TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

ROBOTIC CAPABILITIES IN DISTRIBUTED SYSTEMS: COMMUNICATION, LOCALIZATION, AND RESILIENCE

Jiaqiang Zhang

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SERIAL F 89 — TECHNICA - INFORMATICA— TURKU 2026

University of Turku

Faculty of Technology
Department of Computing
Information and Communication Technology
Doctoral Programme in Technology (DPT)

Supervised by

Professor, Tomi Westerlund
University of Turku

Dr, Xianjia Yu
University of Turku

Docent, Jorge Peña Queralta
University of Turku

Reviewed by

Professor, Jari Nurmi
Tampere University

Dr, Zuoya Liu
Finnish Geospatial Research Institute

Opponent

Professor, Juha Röning
University of Oulu

ISBN 978-952-02-0761-8 (PDF/ONLINE)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)

I dedicate this thesis to my parents for their love and support.

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Information and Communication Technology
ZHANG, JIAQIANG: Robotic Capabilities in Distributed Systems: Communication, Localization, and Resilience
Doctoral dissertation, 65 pp.
Doctoral Programme in Technology (DPT)
April 2026

ABSTRACT

Robotic systems increasingly operate as distributed, multi-host systems spanning robots, edge resources, and cloud services. In such settings, communication, localization, and resilience jointly determine system performance, yet existing research has often treated them separately. This thesis investigates how robotic capability can be strengthened in distributed ROS 2 systems through the combined study of communication architecture, multisensor localization, and resilient edge deployment.

The thesis first analyses communication in the edge–cloud continuum and shows, through a comparative study of CycloneDDS, MQTT, and Zenoh, that middleware performance is conditioned by deployment context: CycloneDDS is most effective in wired Ethernet environments, whereas Zenoh is better suited to Wi-Fi and 4G settings. The dissertation then addresses localization as a multisensor problem. It reviews event-based sensor fusion for odometry, investigates the effect of LiDAR heterogeneity through comparative benchmarking of dome-shaped, solid-state, and spinning lidars, and evaluates seamless outdoor–indoor pedestrian positioning through GNSS/UWB/IMU fusion using error-state Kalman filtering, factor-graph optimization, and particle filtering. The findings show that localization accuracy, robustness, and continuity depend on the interaction between sensor characteristics, estimator structure, and environmental conditions; among the tested pedestrian-positioning back-ends, the error-state Kalman filter yields the most consistent overall performance. Finally, the thesis examines resilience through a Kubernetes-orchestrated ROS 2 multi-robot localization system for UWB-based relative positioning. Fault-oriented experiments demonstrate that container orchestration can sustain localization quality through automated recovery even when edge-side services fail.

The thesis shows that distributed robotic capability is a systems property emerging from the co-design of communication, localization, and resilience. Its contribution is to demonstrate that robust ROS 2 systems depend on aligning communication with network conditions, localization with sensing heterogeneity, and resilience with deployment conditions.

KEYWORDS: Distributed robotic systems, ROS 2, communication middleware, localization, Kubernetes, UWB, GNSS

TURUN YLIOPISTO
Teknillinen tiedekunta
Tietotekniikan laitos
Tietotekniikka
ZHANG, JIAQIANG: Robotic Capabilities in Distributed Systems: Communication, Localization, and Resilience
Väitöskirja, 65 s.
Teknologian tohtoriohjelma
Huhtikuu 2026

TIIVISTELMÄ

Robottijärjestelmät toimivat hajautettuina järjestelminä, jotka ulottuvat roboteista reunalaskentaresursseihin ja pilvipalveluihin. Tällaisissa ympäristöissä viestintä, paikannus ja resilienssi määrittävät suorituskykyä. Aiemmassa tutkimuksessa niitä on tarkasteltu erillään. Tässä väitöskirjassa tutkitaan robottijärjestelmien toimintakyvyn vahvistamista hajautetuissa ROS 2 -järjestelmissä tarkastelemalla viestintäarkkitehtuuria, monianturista paikannusta ja vikasietoista reunalaskentaa.

Väitöskirjassa analysoidaan viestintää reuna-pilvi-jatkumossa. CycloneDDS:ää, MQTT:tä ja Zenohia vertailemalla osoitetaan, että suorituskyky määräytyy käyttöympäristön mukaan: CycloneDDS toimii tehokkaimmin Ethernet-ympäristöissä, kun taas Zenoh soveltuu Wi-Fi- ja 4G-ympäristöihin. Työssä tarkastellaan tapahtumapohjaista anturifuusiota odometriassa, tutkitaan lidar-antureiden heterogeenisyyttä vertailemalla kupolimuotoista, puolijohde- ja pyörivää lidaria sekä arvioidaan saumatonta ulko-sisätilapaikannusta fuusioimalla GNSS-, UWB- ja IMU-mittauksia virhetila-Kalman-suodatuksen, faktorikuvaajaan perustuvan optimoinnin ja partikkelisuodatuksen avulla. Tulokset osoittavat paikannuksen tarkkuuden, häiriönsietokyvyn ja jatkuvuuden riippuvan antureiden ominaisuuksien, estimaattorin rakenteen ja ympäristöolosuhteiden yhteisvaikutuksesta; testatuista jalankulkijapaikannuksen menetelmistä virhetila-Kalman-suodatin tuottaa johdonmukaisimman kokonaissuorituskyvyn. Lopuksi väitöskirjassa tarkastellaan resilienssiä Kubernetes-orkestroidun ROS 2 -pohjaisen monirobottipaikannusjärjestelmän avulla, jossa suhteellinen paikannus perustuu UWB-teknologiaan. Vikatilannekokeet osoittavat, että konttiorkestrointi kykenee ylläpitämään paikannuksen laatua automaattisen toipumisen avulla, kun reunalaitteiden palvelut vikaantuvat.

Väitöskirja osoittaa, että hajautettujen robottijärjestelmien toimintakyky on järjestelmätason ominaisuus, joka syntyy viestinnän, paikannuksen ja resilienssin yhteissuunnittelusta. Se osoittaa, että häiriönsietokykyiset ROS 2 -järjestelmät edellyttävät viestinnän sovittamista verkko-olosuhteisiin, paikannuksen sovittamista anturien heterogeenisyyteen sekä resilienssin sovittamista käyttöönottokontekstiin.

ASIASANAT: hajautetut robottijärjestelmät, ROS 2, viestintäväliohjelmisto, paikannus, Kubernetes, UWB, GNSS

Acknowledgements

I feel deeply privileged to have pursued my doctoral studies in a supportive and collaborative research environment at the TIERS group, University of Turku. This journey has been enriched not only by the academic resources and opportunities made available to me, but also by the many remarkable people whose guidance, collaboration, and encouragement have shaped this dissertation and my development as a researcher.

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Tomi Westerlund, for the invaluable guidance, support, and encouragement provided throughout my doctoral studies. His expertise, insightful feedback, and trust in my work have been instrumental in shaping both this dissertation and my academic growth. I am also sincerely grateful to my supervisor, Dr. Xianjia Yu, for his continuous support, advice, and generosity in sharing his knowledge over the years. I consider myself fortunate to have had the opportunity to work with a dedicated mentor. Also thanks to Docent Jorge Peña Queralta for providing much help in my early PhD years. I would further like to extend my appreciation to Professor Yuwei Chen, who offered valuable guidance. I am also thankful to my Master's supervisor, Professor Fansheng Chen, whose advice and thoughtful explanations were always of great help whenever I needed direction.

I warmly thank my co-authors and colleagues, Hasier, Paola Torrico Moron, Sahar Salimpour, Salma Salimi, Farhad Keramat, Widhi Atman, Mengya Xu, Haizhou Zhang, Ali Salmasi, and Teemu Saukkio, for their valuable contributions, discussions, and support. Their collaboration has greatly enriched this work. I also wish to acknowledge Professor Wallace Moreira Bessa, Qingqing Li, Gabriel Da Silva Lima, Yasir Al-Ameri, Kimmo Paldanius, Minh Nguyen, and other team members for their expertise, cooperation, and assistance.

My sincere thanks go to the pre-examiners, Professor Jari Nurmi and Dr. Zuoya Liu, for their careful review of this dissertation and for their valuable comments and suggestions. I am equally grateful to Professor Juha Röning for agreeing to act as the opponent at my doctoral defence.

Finally, I am deeply grateful to my parents and family for their love, encouragement, and constant presence in my life, even across distance. This dissertation would not have been possible without their support.

07.04.2026
Jiaqiang Zhang

JIAQIANG ZHANG

Jiaqiang Zhang holds a Master of Science in Circuits and Systems from the Shanghai Institute of Technical Physics of the Chinese Academy of Sciences, Shanghai, P.R.China, obtained in 2020. Since April 2022, Jiaqiang has been a doctoral researcher at the Turku Intelligent and Embedded Robotic System (TIERS) Lab, Department of Computing, University of Turku, Finland. Previously, he worked as a software engineer in Shanghai, China. His research interests include robotics, multi-modal sensing and perception, sensor fusion, and remote sensing.

Table of Contents

Abbreviations

APE	Absolute Pose Error
CDF	Cumulative Distribution Function
CI/CD	Continuous Integration and Continuous Deployment
CPU	Central Processing Unit
DDS	Data Distribution Service
DL	Deep Learning
EKF	Extended Kalman Filter
ENU	East-North-Up
ESKF	Error-State Extended Kalman Filter
ESS	Effective Sample Size
FGO	Factor Graph Optimization
FoV	Field of View
GNSS	Global Navigation Satellite System
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HW	Hardware
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IoT	Internet of Things
K3S	Lightweight Kubernetes
LAN	Local Area Network
LiDAR	Light Detection and Ranging
LIO	LiDAR-Inertial Odometry
LSTM	Long Short-Term Memory
MN	Master Node
MoCap	Motion Capture
MQTT	Message Queuing Telemetry Transport
PDR	Pedestrian Dead Reckoning
PF	Particle Filter
PTP	Precision Time Protocol
QoS	Quality of Service
RAM	Random Access Memory

RMW	ROS Middleware
RMSE	Root Mean Square Error
ROS	Robot Operating System
ROS 2	Robot Operating System 2
RTK	Real-Time Kinematic
SLAM	Simultaneous Localization and Mapping
SROS2	Secure ROS 2
STD	Standard Deviation
UAV	Uncrewed Aerial Vehicle
UWB	Ultra-Wideband
VM	Virtual Machine
WN	Worker Node

List of Original Publications

This dissertation is based on the following original publications, which are referred to in the text by their Roman numerals:

I Jiaqiang Zhang, Farhad Keramat, Xianjia Yu, Daniel Montero Hernández, Jorge Peña Queralta, and Tomi Westerlund. Distributed robotic systems in the edge-cloud continuum with ROS 2: a review on novel architectures and technology readiness. In Proceedings of the 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC), 2022: 1–8.

II Jiaqiang Zhang, Xianjia Yu, Sier Ha, Jorge Peña Queralta, and Tomi Westerlund. Comparison of Middlewares in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS 2 Systems. Journal of Intelligent & Robotic Systems, 2024; 110:162.

III Jiaqiang Zhang, Xianjia Yu, Ha Sier, Haizhou Zhang, and Tomi Westerlund. Event-based Sensor Fusion and Application on Odometry: A Survey. In Proceedings of the 2025 IEEE 6th International Conference on Image Processing, Applications and Systems (IPAS), 2025: 1–6.

IV Doumegna Mawuto Koudjo Felix, Xianjia Yu, Jiaqiang Zhang, Sier Ha, Zhuo Zou, and Tomi Westerlund. Understanding Lidar Variability: A Dataset and Comparative Study Featuring Dome-Shaped, Solid-State, and Spinning Lidars. IEEE Robotics and Automation Letters, 2026; 11(1):570–577.

V Jiaqiang Zhang, Xianjia Yu, Sier Ha, Paola Torrico Morón, Sahar Salimpour, Farhad Keramat, Haizhou Zhang, and Tomi Westerlund. Seamless Outdoor-Indoor Pedestrian Positioning System with GNSS/UWB/IMU Fusion: A Comparison of EKF, FGO, and PF. Accepted by the 17th International Conference on Ambient Systems, Networks and Technologies, 2026

VI Jiaqiang Zhang, Xianjia Yu, and Tomi Westerlund. Enhancing the Resilience of ROS 2-Based Multi-Robot Systems with Kubernetes: A Case Study on UWB-Based Relative Positioning. Sensors, 2025; 25(16):5067.

List of Co-authored Publications

The list of co-authored publications that are not included in this dissertation is as follows:

I Muhammad Farhan Humayun, Ali Salmasi, Jiaqiang Zhang, Tomi Westerlund, and Jukka Heikkonen. A Robust Integrated Approach for Near Real-Time Seamless Orthomosaic Generation Using Off-the-Shelf UAVs for Ultra-High Resolution Mapping Applications. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2025; X-2/W2-2025: 65–72.

Declaration of AI use

In preparing this dissertation, generative AI tools (ChatGPT and Grammarly) were used for grammar, polishing, spelling, readability, and LaTeX formatting.

1 Introduction

1.1 Background and Motivation

Robotic systems are increasingly deployed as connected, networked, and computationally distributed systems rather than as isolated machines. This shift has been described through the overlapping paradigms of cloud robotics, edge robotics, and the edge–cloud continuum [1; 2], all of which seek to extend robotic capability beyond the physical platform by distributing computation, storage, coordination, and supervision across heterogeneous infrastructure [3; 4; 5; 6; 7]. In such settings, the robot is no longer a self-contained unit. It becomes a participant in a wider computational system whose behavior depends on communication performance, middleware design, deployment strategy, and the availability of external resources [8].

Within this transition, Robot Operating System 2 (ROS 2) has emerged as the principal middleware framework for contemporary robotic software. Its design emphasizes distribution, modularity, interoperability, and product readiness, while its reliance on the Data Distribution Service (DDS) [9] gives it a data-centric communication model suitable for multi-process and multi-host robotic applications [10; 11]. Yet the move from single-host robotics to distributed ROS 2 systems also introduces new engineering and scientific challenges. Message delivery must remain reliable across heterogeneous networks; nodes may need to migrate between robot, edge, and cloud resources; and increasingly complex software stacks must be deployed and maintained across devices with very different computing capabilities. Communication in distributed robotics therefore cannot be reduced to a background implementation detail. It is a constitutive part of robotic capability itself.

This is particularly evident when robotic systems operate across wireless and resource-constrained environments. A communication approach that performs well in a wired laboratory setting may degrade substantially under Wi-Fi or cellular connectivity, with direct consequences for the timeliness of control, sensing, and inter-robot coordination. The literature has shown that communication behavior in ROS 2 depends not only on middleware choice, but also on discovery overhead, message size, network topology, and the relation between local and remote computation [12; 13; 14; 15]. For distributed robotic systems, the question is therefore not merely how to exchange data, but how to do so in a manner that preserves system-level efficiency and reliability under realistic deployment conditions.

If communication is one enabling layer of distributed robotics, localization is

another. Robotic systems must estimate position, motion, and relative spatial relationships in environments that are often dynamic, degraded, or only partially observable. Over the last decade, localization research has increasingly moved toward multisensor fusion, combining complementary sensing modalities rather than relying on any single sensor in isolation [16; 17]. Vision-inertial systems, Light Detection and Ranging(LiDAR)-inertial odometry, and semantic or map-assisted estimators all illustrate that localization performance depends on the interaction between sensing physics, estimation architecture, and environmental structure [18; 19; 20; 21; 22]. This is even more pronounced in distributed systems, where local motion estimation, shared observations, and external infrastructure may all contribute to the final position estimate.

Sensor heterogeneity is central to this problem. Event cameras provide high temporal resolution, low latency, and strong performance under difficult lighting or fast motion, making them attractive complements to conventional cameras and inertial sensors [23; 24; 25]. LiDAR sensors differ substantially in field of view, scan geometry, point density, and environmental suitability, which means that the behavior of odometry and Simultaneous Localization and Mapping (SLAM) pipelines can change materially when the sensing platform changes. Global Navigation Satellite System (GNSS) offers global positioning outdoors but degrades in occluded or urban environments; Ultra-Wideband (UWB) provides accurate local ranging indoors and in GNSS-denied settings but remains sensitive to non-line-of-sight effects and infrastructure layout [26; 27]; inertial and pedestrian dead reckoning methods provide continuity but accumulate drift [28; 29; 30; 31; 32; 22]. The motivation for studying localization in distributed systems therefore lies not only in improving accuracy, but in understanding how heterogeneous sensors can be fused to achieve robustness and continuity across changing environments.

A third challenge follows directly from the first two: resilience. As robotic capability becomes distributed across software nodes, communication channels, and edge resources, failure can no longer be treated as an exceptional event confined to a single component. Delayed communication, unavailable services, degraded sensor processing, or failed containers may all propagate through the system and impair perception, coordination, and localization. Recent work on containerized robotic software, Kubernetes-based orchestration, and mixed-criticality deployment has shown that resilience in robotics increasingly depends on how computation is packaged, scheduled, monitored, and recovered across the infrastructure that supports the robot [33; 34; 35; 36; 37; 38]. This is particularly significant in cooperative localization and multi-robot systems, where the failure of one processing node may affect the information available to all others [39]. Resilience is therefore not merely a property of control or estimation algorithms; it is also a property of system architecture and deployment.

The motivation of this thesis arises from the intersection of these three themes.

Communication, localization, and resilience are often examined separately, yet in distributed robotic systems they are deeply interdependent. Communication architecture determines how sensing and coordination are made available across the system. Localization determines whether the system can act coherently in space, especially under heterogeneous sensing conditions. Resilience determines whether these capabilities can be maintained when the operating environment, network conditions, or software infrastructure become adverse. The present dissertation approaches robotic capability from precisely this integrated perspective.

1.2 Research Objectives

The overall objective of this dissertation is to investigate how robotic capability can be strengthened in distributed systems through the joint study of communication architecture, localization under heterogeneous sensing conditions, and resilient edge deployment. Rather than treating these as isolated technical domains, the thesis examines how they interact in ROS 2-based robotic systems that operate across multiple hosts, multiple sensors, and multiple layers of infrastructure.

More specifically, the thesis pursues three interrelated objectives. First, it seeks to clarify how architectural choices and communication middlewares affect the efficiency and reliability of distributed ROS 2 systems in edge-to-edge and edge-to-cloud settings. Second, it seeks to examine how heterogeneous sensor properties and multisensor fusion strategies shape localization accuracy, robustness, and continuity across diverse operating environments. Third, it seeks to determine how container orchestration at the edge can improve the resilience of distributed robotic localization applications when individual services fail. Together, these objectives define the dissertation's central concern: how to design distributed robotic systems whose communication, estimation, and recovery mechanisms support one another rather than fail independently.

1.3 Research Questions

The thesis is guided by the following research questions:

RQ1. How do architectural and communication middleware choices shape communication efficiency and reliability in distributed ROS 2 robotic systems across edge-to-edge and edge-to-cloud deployments?

RQ2. How do heterogeneous sensor characteristics and multisensor fusion strategies jointly determine localization accuracy, robustness, and continuity in robotic systems across diverse environments?

RQ3. How can Kubernetes-orchestrated ROS 2 edge deployment enhance the resilience of UWB-based multi-robot relative localization?

Taken together, these questions move from system architecture to estimation and

then to operational continuity. They are designed to capture the central proposition of the thesis: that robotic capability in distributed systems emerges from the interaction of communication, localization, and resilience rather than from advances in any one of these dimensions alone.

1.4 Structure of the Thesis

This dissertation is a compilation thesis of six scientific articles, referred to as Publications I–VI throughout the dissertation. Publication I establishes the architectural background of ROS 2-based edge–cloud robotics. Publication II evaluates communication middleware in distributed ROS 2 deployments. Publications III–V address localization from event-based sensing, LiDAR variability, and GNSS, UWB, and Inertial Measurement Unit (IMU) fusion perspectives. Publication VI investigates resilience through Kubernetes-based orchestration in a ROS 2 multi-robot system.

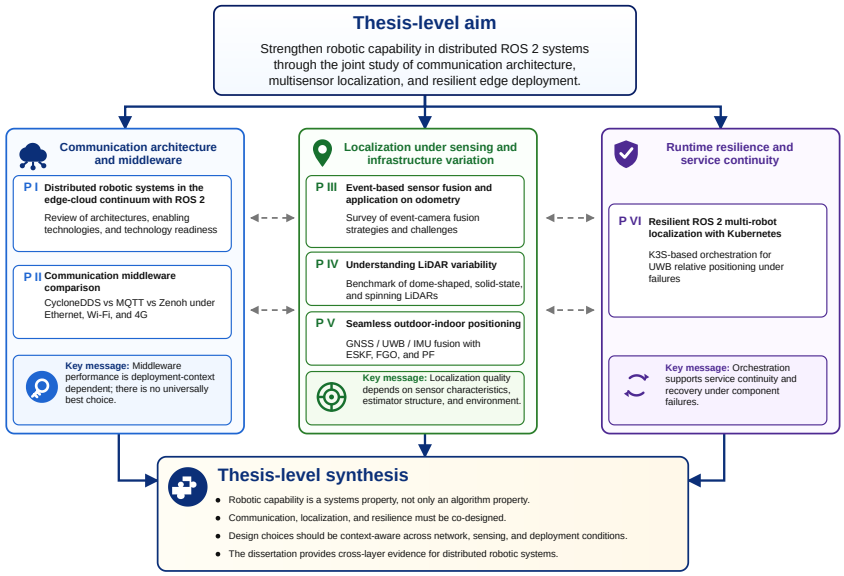


Figure 1. Overview of the research framework and main contributions.

- Chapter 1 introduces the background, objectives, research questions, and overall structure of the study.
- Chapter 2 addresses communication architecture in distributed robotic systems. It begins by examining the broader ROS 2 edge–cloud context and then analyzes communication performance through a comparative study of networking middlewares in multi-host ROS 2 deployments.

- Chapter 3 focuses on localization in distributed robotic systems. It discusses event-based sensor fusion for odometry, investigates the effect of LiDAR heterogeneity on SLAM and point-cloud registration, and examines seamless outdoor–indoor positioning through GNSS/UWB/IMU fusion.
- Chapter 4 addresses resilience in distributed robotic systems through a study of Kubernetes-orchestrated ROS 2 edge deployment for UWB-based multi-robot relative localization.
- Chapter 5 reviews the six publications that form the research core of the dissertation and discusses their scientific contribution as well as the doctoral candidate's contribution.
- Chapter 6 concludes the thesis by synthesizing the main findings and outlining directions for future work.

2 Architectural Foundations and Communication Middleware in Distributed Robotic Systems

This chapter synthesizes the architectural and communication-related findings of Publications I and II. Publication I provides the broader edge–cloud robotics background, whereas Publication II presents the empirical middleware comparison across Ethernet, Wi-Fi, and 4G communication settings.

2.1 Concepts of Distributed Robotic Systems

2.1.1 From standalone robots to distributed robotic systems

Robotic systems are more connected, networked, and distributed than ever. There is a growing intersection between the Internet of Things and robotics, and significant potential can be found at the intersection of autonomous robots with edge and cloud computing. This development is leading to a wide range of new architectures, technologies, and system proposals through which robots can leverage the edge-cloud computing continuum to enhance autonomy [40], support multi-robot cooperation and human-robot collaboration [41; 42; 6], and exploit additional computational capabilities [3; 4; 5; 8].

The benefits of wider adoption of edge and cloud computing paradigms in robotics are considerable. However, these benefits are not obtained simply by attaching network connectivity to a robot. They require systematic architectural support, deeper integration with the software tools used in robotics, and communication mechanisms that remain effective when robotic functions are distributed across several hosts. In this sense, distributed robotic systems are not merely collections of connected devices. They are robotic systems in which sensing, computation, data exchange, and coordination are explicitly organised across multiple nodes.

The move from standalone robots to distributed robotic systems is therefore both a technological and an architectural shift. Earlier systems often assumed that perception, planning, and control were executed on a single host or within a tightly bounded robotic platform. Contemporary systems increasingly rely on multiple computers, remote services, shared infrastructure, and multi-robot interaction. In such systems, communication is not only a support mechanism but a constitutive part of the robotic

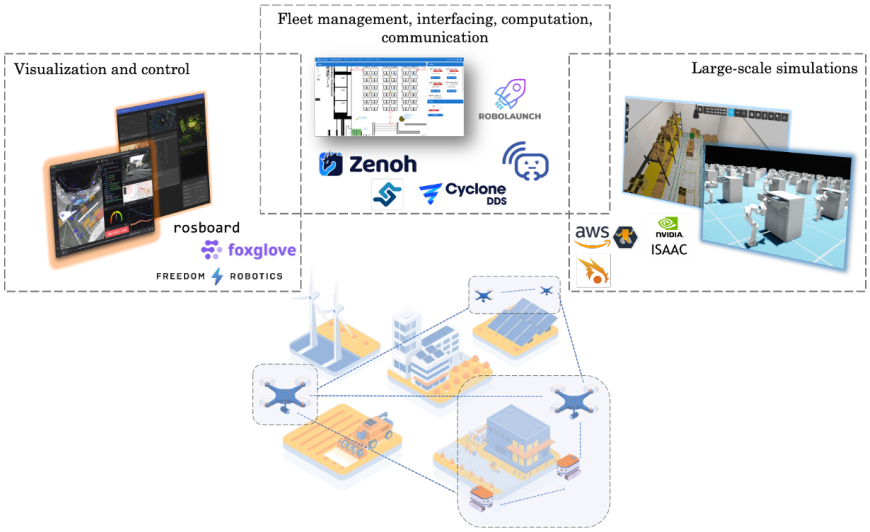


Figure 2. Illustration of a subset of edge-fog-cloud robotic platforms and tools, and the potential areas where cloud and edge computing can play a key role for autonomous robots. From visualization and teleoperation to simulation and online learning, and including fleet management or computational offloading, connectivity can be leveraged to build more robust and capable distributed robotic systems.

capability itself.

This shift also changes how robotic capability should be understood. A capability such as perception, localization, mapping, task allocation, or supervision may now emerge from the coordinated operation of onboard components, edge servers, external services, and user-facing tools. The problem is therefore no longer limited to selecting algorithms; it also includes deciding where functions are executed, how information is exchanged, and how distributed operation remains manageable under realistic network conditions.

2.1.2 Edge-fog-cloud Continuum and Crchitectural Requirements

The architectural context of distributed robotic systems is commonly described as an edge-fog-cloud continuum [1; 2]. In robotics, the edge usually refers to the robot and the computing resources in its immediate environment, the cloud refers to remote infrastructure with larger computational and storage capacity, and fog computing occupies the intermediate space by bringing lower-latency resources closer to the robot [43]. These layers should not be treated as isolated categories, but as a continuum across which robotic functions are placed according to latency, bandwidth, and resource requirements.

This continuum enables remote visualisation and teleoperation, simulation, online learning, fleet management, and computational offloading [44; 45; 46; 47; 48;

49; 50; 51]. In practice, most distributed robotic systems are hybrid rather than purely local or purely cloud-based. Latency-sensitive functions such as sensing, estimation, and control usually remain close to the robot, while more compute-intensive or service-oriented workloads may be moved to nearby or remote infrastructure [7; 52; 53]. The architectural implication is that distributed robotic systems must remain modular, support heterogeneous hardware and networking conditions, and preserve acceptable performance outside ideal laboratory settings.

2.1.3 ROS 2 as a Software and Communication Foundation

ROS 2 provides the most relevant software foundation for this type of distributed robotics. It has become the de facto standard in robotics research and development, while offering improved support for real-time communication, modular software decomposition, and multi-host deployment [10]. Its design principles include distribution, abstraction, asynchrony, and modularity, together with support for embedded systems, diverse networking conditions, real-time computing, and product readiness. These properties make ROS 2 not only a development framework, but also an architectural substrate for robotic systems whose functionality is distributed across nodes, hosts, and infrastructure layers.

A typical ROS 2 system is composed of distributed nodes that communicate through topics, services, and actions. This computation model allows robotic functionality to be decomposed into reusable and deployable components while preserving a coherent programming interface across single-host and multi-host systems. At the communication layer, ROS 2 relies on DDS through the ROS middleware abstraction. DDS provides a data-centric communication model and Quality of Service mechanisms for expressing reliability, latency, and throughput requirements, while the abstraction layer separates application logic from the underlying transport implementation [9; 11; 10]. At the same time, some data-intensive robotic workloads, especially those involving high-quality images or dense point clouds, may require complementary communication or streaming mechanisms beyond DDS alone [11]. For this reason, ROS 2 should be understood as the common architectural and communication foundation of distributed robotic systems, rather than as the final answer to every deployment problem.

2.2 Deployment Enablers for Distributed Robotic Systems

2.2.1 Containerization

Virtualization technologies are among the core foundations of cloud computing. While virtual machines traditionally formed the backbone of cloud infrastructure,

container technology has emerged as a lightweight alternative. Containers provide an environment in which applications can run together with the dependencies they require, thereby simplifying deployment across heterogeneous platforms [54].

Docker has become one of the dominant containerization platforms. In robotics, containerization is valuable because it improves portability, reproducibility, and consistency across different devices [55; 56]. A ROS 2 application that has been containerized can be deployed more systematically on development machines, edge devices, and cloud hosts.

For distributed robotic systems, this is more than a convenience. Containerization supports modular deployment and makes it easier to manage software components whose functions are already separated at the ROS 2 node level. It therefore acts as an important practical enabler of distributed robotic architectures.

2.2.2 Orchestration

As containerized applications become distributed across multiple devices, orchestration becomes necessary. Kubernetes is a platform for orchestrating containers, and lightweight Kubernetes distributions such as K3S are especially relevant for edge-side deployments. Kubernetes automates deployment, scaling, and management, thereby providing a structured way to operate containerized applications across distributed systems [57].

In the robotics literature, Kubernetes and K3S (Lightweight Kubernetes) have been used to support cloud-edge deployment of robotic applications, to manage containerized simulation, and to orchestrate ROS-based software across hierarchical and heterogeneous hardware architectures. These developments show that orchestration is increasingly becoming part of the practical deployment stack for distributed robotics [58; 34; 35; 36; 59].

In the context of this chapter, orchestration is treated as a deployment enabler rather than as a resilience solution in itself. Its relevance here lies in the fact that it makes distributed ROS 2 systems practically deployable across the edge-cloud continuum. Questions of runtime continuity and application-specific fault handling are treated later in the thesis.

2.3 Communication Middleware for Distributed ROS 2 Systems

2.3.1 Communication requirements in distributed ROS 2 systems

Efficient and reliable communication protocols for distributed systems are in significant demand with the development of the Internet of Things (IoT) and edge com-

puting. In robotics and autonomous systems, the wide use of ROS 2 brings the possibility of utilising various networking middleware solutions together with DDS in ROS 2 for better communication among edge devices or between edge devices and the cloud.

The communication problem is intensified by the large amount of data generated in the system, the heterogeneous devices involved, and the diversity of networking conditions. In multi-host ROS 2 systems, communication increasingly takes place over Ethernet, Wi-Fi, or 4G rather than only within a single machine [60; 61]. Under these conditions, latency, throughput, reliability, and message characteristics become key design concerns.

A meaningful communication evaluation must therefore consider not only a single middleware but a set of middleware options under several network regimes and with different ROS message types and sizes. This is the basis of the comparative study discussed in the remainder of this section.

2.3.2 DDS and CycloneDDS

DDS is the default communication middleware of ROS 2 and remains the basis of the ROS middleware abstraction. It provides a decentralized architecture in which nodes can communicate directly with each other without relying on a centralized broker. This architecture, shown in Fig. 3, supports reliability, fault tolerance, and scalability, and includes Quality of Service (QoS) settings that allow applications to specify their communication needs [9; 62].

In the middleware comparison considered in this chapter, CycloneDDS is used as the DDS implementation. The focus is not on comparing all DDS implementations, but on understanding how a representative DDS-based ROS 2 system performs in multi-host communication scenarios.

CycloneDDS performs particularly well under Ethernet conditions. This is consistent with the fact that DDS was originally designed to be more suitable for Ethernet environments than for open or wireless environments. In wired local networks, multicast-based discovery and peer-to-peer communication can support low-latency message exchange efficiently [61].

2.3.3 Message Queuing Telemetry Transport (MQTT)

MQTT is a lightweight publish-subscribe protocol designed for constrained devices and low-bandwidth network environments and has become almost a de facto standard in the field of IoT. It supports different QoS levels and is widely used for remote device communication and machine-to-machine interaction [63; 64; 65].

Within distributed ROS 2 systems, MQTT is not a replacement for ROS 2 itself. Rather, it is a networking middleware that can be combined with ROS 2 in multi-host

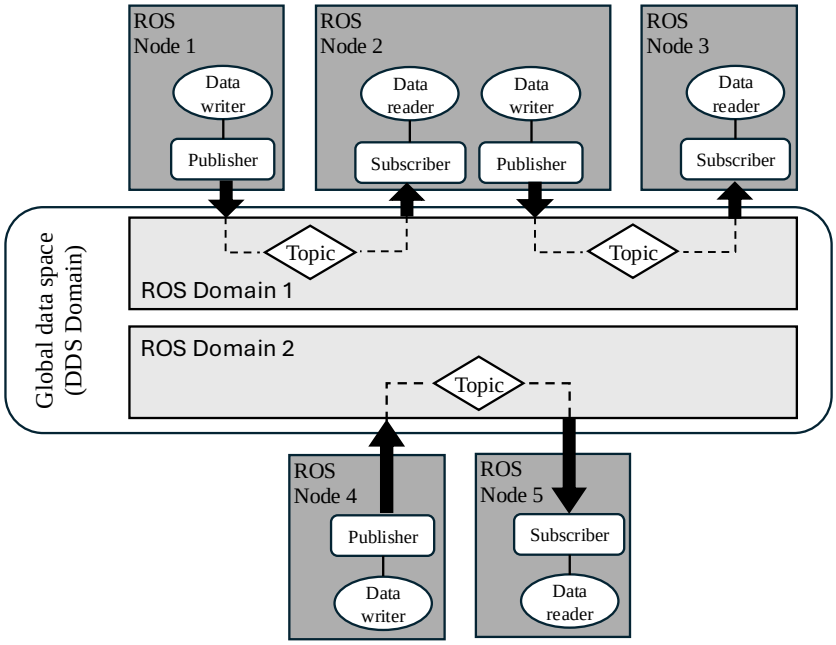


Figure 3. DDS architecture in the ROS 2 system

deployments. This makes it an interesting candidate for edge-to-edge and edge-to-cloud communication, especially where lower network overhead is desirable.

The architectural logic of MQTT differs from DDS because it is associated more naturally with broker-based messaging patterns. This gives it certain practical advantages in some IoT-oriented settings, but it also changes the communication topology and dependency structure of the robotic system.

2.3.4 Zenoh

Zenoh is an open-source protocol and suite of tools for data sharing and communication in distributed systems. It aims to provide a unified approach to data sharing and communication regardless of the underlying hardware, network topology, or programming language. This makes it particularly relevant to distributed robotics in the edge-cloud continuum [66].

Zenoh provides a data-centric communication model and supports edge-to-edge, edge-to-cloud, and hybrid environments. In the context of ROS 2 systems, its appeal

lies in its ability to support discovery-efficient publish-subscribe communication under network conditions in which DDS alone may be effective [15].

Because Zenoh is designed with heterogeneous and distributed deployment in mind, it is especially suitable for empirical comparison with DDS in multi-host ROS 2 systems. This is why it is treated in the middleware paper as a serious communication alternative rather than merely a theoretical possibility.

2.3.5 Integration models for multi-host ROS 2 communication

A key architectural aspect of the middleware comparison is that DDS remains the ROS 2 middleware within each host in all experiments. In the Zenoh and MQTT experiments, localhost-only DDS is enabled, and a bridge is set between DDS and Zenoh or MQTT so that DDS publishes or subscribes ROS messages only to or from the local host. In other words, communication within one host uses DDS, while communication between two hosts uses Zenoh or MQTT.

This hybrid design is important because it reflects a realistic multi-host integration pattern. The study is not comparing ROS 2 against completely separate communication stacks. It is comparing ways of complementing the default ROS 2 middleware with alternative networking middleware for inter-host communication.

Architecturally, this means that middleware selection in distributed ROS 2 systems is often a matter of composition rather than substitution. Local DDS communication can be retained, while inter-host communication can be adapted to the network regime in which the system is deployed.

2.4 Comparative Evaluation of Communication Middleware

2.4.1 Experimental methodology

The comparative evaluation is conducted in a multi-host ROS 2 system under three network environments: Ethernet, Wi-Fi, and a Zerotier virtualized 4G setup. These environments represent, respectively, a high-bandwidth low-latency local regime, a wireless edge-to-edge regime, and an edge-to-cloud-like regime with more restrictive communication conditions. The experiment setup is shown in Fig. 4.

To provide a complete and reliable comparison, the study measures latency and throughput using distinct types and sizes of ROS messages. These include arrays ranging from 1 kB to 2 MB and point cloud messages of comparable size. Messages are published at 10 Hz. Reliable QoS is used to ensure that the collected data reflects communication performance rather than avoidable message loss. The performance tool selected for this research is performance_test [67].

The experimental setup includes two hosts, a router or broker host, the hardware

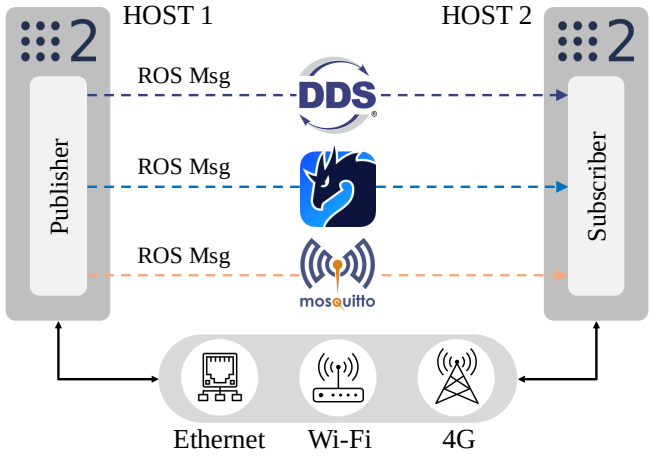


Figure 4. Proposed evaluation scheme to test DDS, Zenoh, and Mosquitto in different network setups

setup shown in Table 1; and a real-robot evaluation with a TurtleBot 4, shown in Fig. 5, and a laptop connected to the same Wi-Fi network. In the robot experiment, the laptop sends velocity commands so that the TurtleBot 4 executes four identical square loops over 96 seconds. The actual trajectory is recorded with an OptiTrack motion-capture system. This allows the evaluation to extend beyond synthetic benchmarking and into an actual robotic task.

Table 1. Hardware Setup

Devices	System	PROCESSOR	Memory
Host 1	Ubuntu 20.04	AMD®Ryzen 7 5800h	16GB
Host 2	Ubuntu 20.04	Intel® Core i3-1215U	16GB
Router/Broker	Ubuntu 20.04	Intel® Core i7-9700K	64GB

2.4.2 Performance under Ethernet

The Ethernet setup, shown in Fig. 6(a), utilized an Ethernet local network, representing a high-bandwidth and low-latency environment for Edge-to-Edge communication. The Local Area Network (LAN) Switch used in Ethernet Local Network is Unifi Flex Mini. Under Ethernet conditions, CycloneDDS performs best overall. Mean latency remains lower than that of Zenoh and MQTT across most tested message sizes, and throughput is likewise strong. This result is consistent with the design

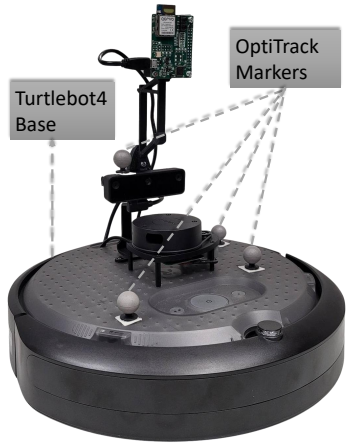


Figure 5. The Turtlebot 4 robot used in the experiment with OptiTrack markers on top

expectations of DDS and with its suitability for Ethernet environments.

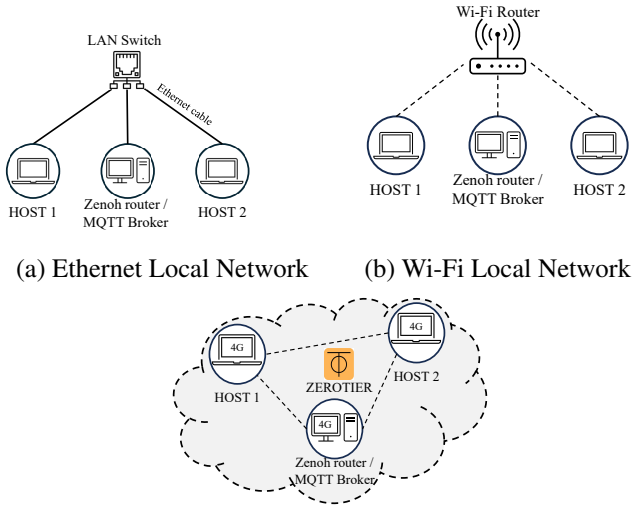
The Ethernet results show that the default ROS 2 communication basis remains highly effective in stable wired local networks. In such settings, direct DDS communication provides a strong baseline and, in the present study, the best performance among the compared approaches.

This result is architecturally important because it shows that alternative middleware should not be adopted indiscriminately. The empirical evidence supports the continued use of DDS/CycloneDDS as the preferred choice in wired local deployments.

2.4.3 Performance under Wi-Fi and 4G

The setups of Wi-Fi and 4G are shown in Fig. 6(b) and (c). The Wi-Fi setup involved a local network using Wi-Fi connectivity, simulating Edge-to-Edge communication within a confined area. The Wi-Fi Router is Huawei B593. Lastly, in 4G setup, we leveraged the Zerotier virtualized network to 4G setup, which emulates Edge-to-Cloud communication scenarios. In this scenario, for HOST 1 & 2, we used Huawei 4G Dongle E3372 as 4G module, and for broker, we used Netgear 4G LTE Mobile Router as 4G module. The results change under Wi-Fi and 4G. In both regimes, Zenoh performs better than CycloneDDS overall, particularly in latency behaviour and throughput stability. This finding is consistent with the observation that DDS discovery protocols may cause flooding effects in open or wireless environments [15; 68; 61].

As message size increases, latency increases for all middleware solutions, but the wireless and 4G setups expose clearer differences among them. Zenoh shows more favourable behaviour than CycloneDDS in these environments, while MQTT



(a) Ethernet Local Network (b) Wi-Fi Local Network

(c) 4G (Zerotier Virtualized Network)

Figure 6. Network setups. The router or broker is for Zenoh and MQTT; DDS does not need an individual broker.

exhibits significantly higher latency penalties, particularly for larger messages and more demanding network conditions.

The main design implication is that the network environment strongly influences middleware performance. CycloneDDS is strongest under Ethernet, while Zenoh is better suited to Wi-Fi and 4G.

2.4.4 Real-robot trajectory drift evaluation

In the actual robot experiment, shown in Fig. 7, the laptop sends velocity commands to the TurtleBot 4 so that it follows a square-shaped trajectory over 96 seconds. The OptiTrack motion-capture (MoCap) system records the actual trajectory, and the drift between the executed path and the intended repeated loops is used as an additional performance indicator.

Zenoh demonstrates the closest adherence to the square trajectory, exhibiting minimal drift and reproducing the desired path more accurately than the other tested middleware configurations. CycloneDDS and MQTT show greater degrees of drift over time. This result highlights the fact that communication middleware affects not only benchmark metrics such as latency and throughput, but also the physical behaviour of the robot.

This is one of the most important findings of the chapter. It shows that middleware choice is not only a backend systems issue. It influences embodied robotic

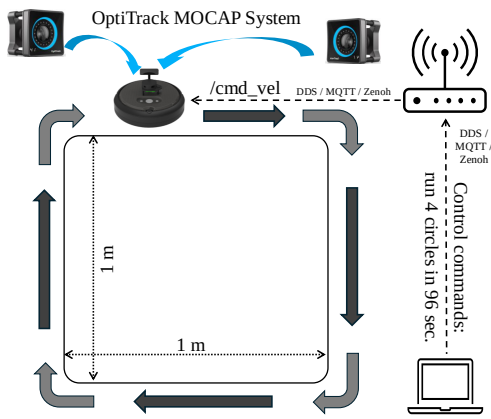


Figure 7. Actual robot setup. The TurtleBot 4 receives the "/cmd_vel" velocity command from the HOST to run a square trajectory. The linear velocity is set to 0.33 m/s forwarding, and the turning velocity is set to 30 degree/s. The OptiTrack MOCAP System is only used to record the trajectories, which are subsequently employed in calculating the drift errors.

performance and should therefore be treated as part of robotic system design rather than as an isolated networking decision.

3 Multisensor Fusion for Localization in Robotic Systems

This chapter brings together the localization-related contributions of Publications III, IV, and V. Publication III reviews event-based sensor fusion for odometry, Publication IV examines LiDAR heterogeneity through a comparative dataset, and Publication V evaluates GNSS/UWB/IMU fusion for seamless outdoor–indoor positioning.

3.1 Multimodal Sensors for Localization

Reliable localization in robotic systems rarely emerges from a single sensing modality. In practical deployments, localization performance is shaped by the complementary strengths and weaknesses of visual, inertial, ranging, and radio-based sensors. Cameras provide rich scene information, but their performance degrades under rapid motion, low illumination, and strong brightness variation. Inertial sensors provide high-rate motion cues, but inertial integration accumulates drift over time. LiDAR offers accurate geometric sensing and is widely used for odometry and mapping in GNSS-denied environments, yet its performance remains sensitive to scene structure, field of view, scan pattern, and sensor configuration. Outdoor and indoor absolute positioning systems, such as GNSS and UWB, provide globally or locally referenced position updates, but both are vulnerable to environmental limitations, including multipath, non-line-of-sight propagation, and partial coverage.

For these reasons, localization in robotic systems is fundamentally a multisensor problem. Sensor fusion allows motion continuity from inertial sensing, geometry from LiDAR, scene dynamics from cameras, and bounded absolute corrections from radio-based positioning to be brought into a unified estimation framework. In this chapter, multimodal sensing is approached from two complementary perspectives. The first is sensor-centred and addresses the role of heterogeneous sensing modalities, event-based odometry, and LiDAR variability in localization. The second is system-centred and examines how GNSS, UWB, and IMU-based pedestrian dead reckoning can be fused to support seamless outdoor–indoor positioning.

3.1.1 Sensor modalities for localization

Robotic localization spans a broad spectrum of tasks, from short-horizon odometry to globally referenced positioning. Odometry estimates the relative motion of the platform over time, typically from visual, inertial, or LiDAR observations. Global localization, by contrast, anchors the platform to an external frame through satellite navigation, infrastructure-based ranging, or map-referenced observations. In realistic robotic systems, these two functions are closely intertwined: odometry provides continuity, while global measurements limit long-term drift.

Visual sensors remain central to robotic perception and localization. Conventional frame-based cameras provide dense spatial information and support a large body of methods for feature extraction, matching, and geometric reconstruction. They are, however, prone to motion blur during rapid motion and perform poorly in scenes with severe illumination changes. Event cameras introduce a different sensing paradigm. Instead of recording frames at fixed intervals, they operate asynchronously and report brightness changes independently at each pixel. This sensing principle provides high temporal resolution, low latency, high dynamic range, and reduced sensitivity to motion blur, making event cameras particularly attractive for fast and challenging odometry tasks.

IMUs provide angular velocity and linear acceleration at high frequency and are therefore indispensable for motion continuity. Their measurements are particularly valuable when visual or geometric observations become intermittent or degraded. Nevertheless, inertial integration alone cannot sustain accurate long-term localization, because small biases and modelling errors accumulate rapidly. For this reason, IMUs are typically fused with visual, LiDAR, GNSS, or UWB observations.

LiDAR has become one of the primary sensors for robot localization in GNSS-denied or unreliable environments. It provides explicit three-dimensional geometric observations of the environment and is widely used in odometry, SLAM, and point cloud registration. At the same time, LiDAR is no longer a single homogeneous sensor category. Spinning LiDARs, solid-state LiDARs, and dome-shaped LiDARs exhibit different fields of view, scan patterns, spatial densities, and operational trade-offs. These differences have a direct influence on localization performance and on the suitability of a given sensor for a particular platform or environment.

GNSS and UWB occupy a different role in the sensing hierarchy. GNSS provides globally referenced outdoor position information but suffers in urban settings from signal blockage, multipath, and degradation near buildings. UWB can offer decimetre-level indoor positioning where anchors are deployed, yet it is sensitive to non-line-of-sight conditions and is limited to instrumented spaces. In practice, neither modality is sufficient on its own for continuous localization across environment transitions. Their value lies in complementing inertial or odometric motion estimates with absolute updates.

From a system-design perspective, heterogeneous sensor modalities should not be viewed as redundant alternatives, but as complementary sources of observability. Cameras capture appearance and motion cues, event cameras emphasise temporal change, IMUs stabilise short-term propagation, LiDAR captures geometry, GNSS provides outdoor absolute reference, and UWB provides local indoor reference. Robust localization in robotic systems therefore depends not only on estimator design, but also on the choice, placement, calibration, and synchronization of multiple sensing modalities.

3.1.2 Event-based sensor fusion for odometry

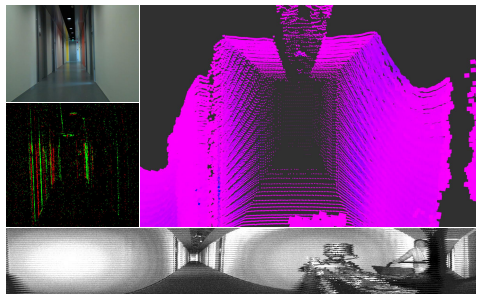


Figure 8. The sensor data representation of a long corridor includes the following: RGB image (top-left), an event-based camera image (middle-left), a LiDAR point cloud (top-right), and a LiDAR-generate reflectivity image (bottom). LiDAR data is adapted from [69].

Event cameras are bio-inspired vision sensors that operate fundamentally differently from traditional frame-based cameras. Instead of capturing images at fixed time intervals, event cameras detect changes in brightness asynchronously, resulting in a stream of events that encodes changes in the scene [70; 71]. This unique sensing modality provides significant advantages in scenarios involving high-speed motion, low lighting, or scenes with significant dynamic range, where traditional cameras may struggle [23].

In the context of robotics and computer vision, sensor fusion involving event cameras has garnered considerable attention, particularly for enhancing visual odometry and visual-inertial odometry. Fig. 8 illustrates the varying representations of a long corridor as captured by different sensors, including an RGB camera, LiDAR, and an event camera. In environments where geometric information is homogeneous, such as long corridors, event camera could enhance the performance of LiDAR odometry, which is prone to drift [72].

An event at a pixel is triggered when the change in logarithmic brightness exceeds a predefined threshold [73]. This sensing mechanism differs fundamentally from the exposure-based acquisition of conventional cameras and leads to four main characteristics that are particularly relevant to odometry. First, event cameras pro-

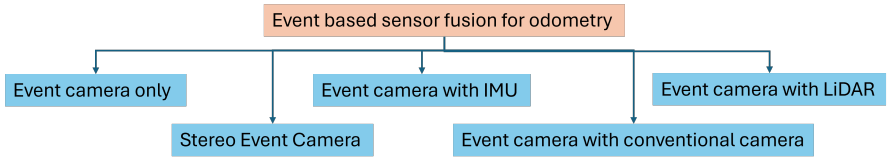


Figure 9. Primary aspects covered in event camera-based fusion for odometry purposes in this survey.

vide high temporal resolution, because they respond immediately to changes in the scene rather than waiting for the next frame interval. Second, they exhibit very low latency, since each pixel operates independently and transmits brightness changes without global exposure. Third, they offer a much higher dynamic range than conventional cameras, allowing them to function reliably in scenes with strong brightness variation. Fourth, event cameras are inherently resistant to motion blur, which is especially beneficial for fast-moving robotic platforms.

These properties make event cameras particularly suitable for sensor fusion. The primary motivation for using event cameras in odometry is their ability to provide continuous, low-latency information that complements the limitations of traditional sensors. Frame-based cameras are prone to motion blur during rapid movements, while IMUs suffer from drift over time. LiDAR-based odometry, although generally more robust to lighting changes, can degrade in environments where geometric structure is weak or repetitive, such as long corridors. Event cameras can provide complementary visual dynamics in such cases and help mitigate these weaknesses.

The earliest event-based pose estimation methods relied on event cameras alone. Early work demonstrated that asynchronous events could support robot self-localization and, later, simultaneous localization and mapping [74; 75]. These studies established the feasibility of using event streams for motion estimation, but the limited spatial structure in event-only representations meant that purely event-based odometry remained comparatively niche.

Fusion with frame-based cameras became a more natural and productive direction [76; 77; 78]. Combining event cameras with frame cameras allows high temporal resolution to be paired with more structured scene information. Event data can reduce blur, recover temporal detail between frames, and improve tracking in dynamic conditions. In odometry, this fusion enables low-latency feature tracking and more robust pose estimation than frame-only methods. Low-latency event-based visual odometry systems demonstrated that asynchronous event streams could be used for real-time odometry in conditions where conventional vision pipelines struggle.

Fusion with IMUs is particularly attractive because the two modalities are naturally complementary [79; 80; 81; 82; 24; 83; 84]. The IMU provides high-rate inertial information that supports motion prediction and short-term continuity, while the event camera contributes visual observations that remain informative under rapid

motion and high dynamic range. Event-based visual-inertial odometry has therefore become one of the most important branches of event-based fusion research. Continuous-time visual-inertial formulations further exploited the asynchronous nature of event data and demonstrated that event-camera and IMU fusion can achieve accurate and robust odometry even in highly dynamic or difficult lighting conditions. More recent work has introduced adaptive filtering and uncertainty-aware schemes to improve resilience to noise and motion irregularities.

Stereo event-camera systems extend the idea further by enabling depth perception through synchronized asynchronous sensing [85; 86; 87; 88; 89; 90; 91; 92; 93; 94]. Event-based stereo visual odometry and stereo visual-inertial odometry have shown that stereo event cameras can support three-dimensional reconstruction and motion estimation in real time. Semi-dense and feature-based methods have both been explored, and recent systems have continued to improve the robustness and practicality of stereo event-based localization.

Fusion with LiDAR remains a relatively early but highly promising direction [95; 96; 97; 98]. LiDAR technology generates precise three-dimensional maps and highly accurate odometry information, but it operates at a lower rate and can struggle when geometric information is homogeneous or partially missing. Event cameras offer a complementary sensing mechanism that may help alleviate these limitations. Initial research has focused strongly on calibration, as accurate extrinsic alignment between event cameras and LiDAR is essential for meaningful fusion. Beyond calibration, event data has also been fused with sparse LiDAR to generate denser depth information. More broadly, event cameras may serve as a bridge between RGB and LiDAR modalities in visual-motion fusion tasks, owing to the homogeneous relationship between event, image, and geometric motion cues.

Recent datasets have supported this growing line of research. Event-camera datasets for odometry, shown in Table 2, now include indoor scenes, driving scenarios, handheld trajectories, and stereo event-camera sequences. Many of these datasets combine event cameras with RGB cameras, IMUs, LiDAR, and GNSS, enabling controlled evaluation of event-based fusion pipelines. Their emergence has contributed substantially to the maturation of the field.

Table 2. Event camera-based dataset related to odometry

Year	Dataset Name	Scenarios	Sensors
2024	CoSEC [99]	Driving scenarios	Event Camera, LiDAR, RGB Camera, IMU, GNSS
2023	ECMD [100]	Driving scenarios	Event Camera, LiDAR, RGB Camera, Infrared Camera, IMU, GNSS
2023	UNIZG-FER LAMOR [89]	Indoor handheld & Outdoor Driving	Stereo Event Camera, LiDAR, IMU, GNSS
2022	Evimo2 [101]	Indoor Scenes	Stereo Event Camera, IMU
2021	DSEC [25]	Driving scenarios	Stereo Event Camera, LiDAR, RGB Camera, IMU, GNSS
2018	MVSEC [102]	Handheld, UAV & Driving	Stereo Event Camera, LiDAR, VI-Sensor, IMU, GNSS
2017	DAVIS Datasets [103]	High-speed robotics	Event Camera, RGB Camera, IMU

From the perspective of robotic localization, the main significance of event-based sensing lies not in replacing established sensing modalities, but in extending their op-

erating envelope. Event cameras are especially valuable when rapid motion, low light, and strong brightness variation challenge conventional imaging. In visual odometry or visual-inertial odometry, they help mitigate motion blur and improve temporal responsiveness. In LiDAR-related localization, they have the potential to compensate for the loss of geometric information by providing complementary photometric cues. Their role in multisensor fusion is therefore best understood as that of an enabling modality that strengthens the continuity and robustness of odometry in demanding environments.

3.1.3 Multi-modal sensor platform and LiDAR variability

While event-based odometry highlights the value of heterogeneous visual sensing, a similar argument applies within LiDAR itself. LiDAR is often treated as a single modality in the localization literature, yet recent developments show that different LiDAR types can exhibit substantially different localization behaviour. This is particularly important in robotic systems, where cost, form factor, field of view, and power consumption often constrain sensor selection as strongly as raw performance.

LiDAR technology has seen significant advancements in recent years, leading to the development of both high-end spinning LiDARs and more affordable solid-state options [104; 21]. Traditional spinning LiDARs, such as those from Velodyne and Ouster, continuously scan the environment with a rotating mechanism and are widely used due to their accuracy and wide Field of View (FoV) [105]. Solid-state LiDARs, by contrast, employ no moving parts and instead rely on alternative beam steering or non-repetitive scanning mechanisms, resulting in more compact and robust designs. Among these, Livox sensors have received increasing attention due to their cost-effective construction and distinctive scan patterns [106]. The Livox Avia provides a limited FoV, while the Livox Mid-360 features a dome-shaped design with a 360°horizontal FoV and a vertical span of approximately 25°. This design makes the Mid-360 particularly attractive for portable mapping systems, unmanned aerial vehicles (UAV), and GNSS-denied environments [107].

These developments motivate a platform-level view of multimodal localization. A synchronized multi-sensor platform makes it possible to compare sensing modalities fairly and to investigate how sensor-specific characteristics affect localization performance. In the present context, a representative platform consists of multiple LiDARs mounted on the same robotic base together with additional sensors such as an RGB-D camera and an IMU. Such a design allows data from different LiDAR categories to be collected under identical motion and environmental conditions. This is particularly valuable because many existing datasets focus mainly on spinning LiDAR systems, while datasets that combine spinning, solid-state, and dome-shaped LiDARs on the same platform remain rare [108; 109; 110; 111; 112; 113].

A platform-centred dataset design also enables more systematic evaluation of

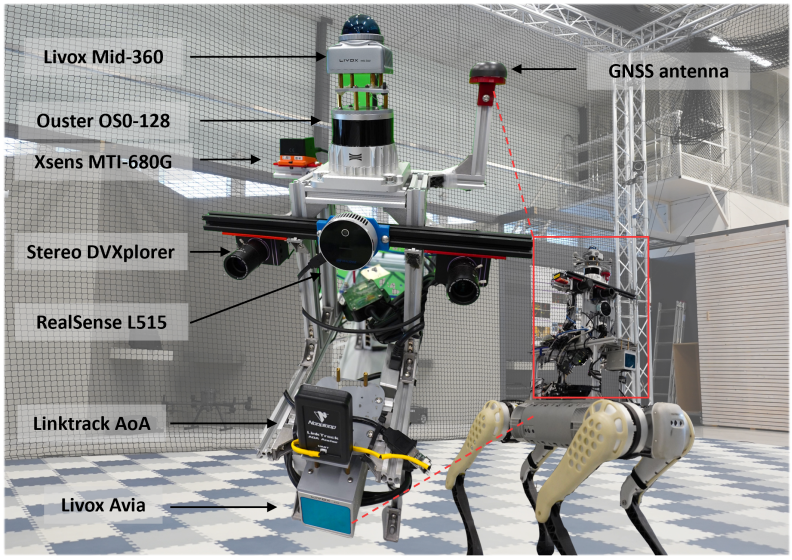


Figure 10. The hardware used for the data collection.

calibration and synchronization. In one such configuration, shown in Fig 10, the moving platform is a Unitree B1 quadruped robot equipped with Livox Mid-360 and Avia, Ouster OS0-128, RealSense L515, DVXplorer, Linktrack AoA, and Xsens MTi-680G sensors. Point clouds from each LiDAR are first approximately aligned through manual measurements and subsequently refined through optimization-based registration, such as Generalized Iterative Closest Point (ICP). Temporal synchronization can be achieved using the Precision Time Protocol when hardware triggering is unavailable, while running sensor drivers and data-logging programs locally on a high-performance computer helps reduce additional latency. Such calibration and synchronization procedures are essential for meaningful cross-sensor comparison.

The importance of LiDAR variability becomes especially clear once data are collected across diverse environments. Indoor scenes typically emphasise structured geometry, shorter sensing ranges, and fine spatial detail, whereas outdoor scenes introduce mixed natural and built structures, occlusions, and much longer-range observations. In one dataset design, indoor sequences are collected inside a controlled factory-like facility with motion-capture ground truth, while outdoor sequences are gathered in road-like and forest-neighbourhood environments with GNSS/Real-Time Kinematic (RTK)-supported ground truth. This pairing of indoor and outdoor data is crucial, because the same LiDAR characteristics that are advantageous in one environment may become less effective in another.

The dataset design also supports two complementary types of localization analysis. The first is LiDAR-based odometry and SLAM benchmarking. Here, multiple state-of-the-art pipelines can be evaluated across identical sequences collected

from different LiDARs. This makes it possible to examine how different scan patterns, fields of view, and spatial densities interact with distinct SLAM architectures. Methods such as FAST-LIO2, FASTER-LIO, S-FAST-LIO, FAST-LIO-SAM, and GLIM cover direct scan-to-map odometry, lightweight variants, loop-closure-enhanced pipelines, and globally optimized range-inertial frameworks. Even without entering the result discussion, such benchmarking establishes an essential methodological point: localization behaviour depends on both the sensor and the odometry pipeline, not just on one or the other in isolation.

The second analysis concerns point cloud matching in IMU-free settings. Modern LiDAR odometry is often tightly coupled with inertial sensing, which can obscure the independent influence of LiDAR geometry on scan registration. To address this, point cloud matching can be evaluated separately using classical ICP variants. Point-to-point ICP minimizes Euclidean distances between corresponding points and is computationally simple, but it is sensitive to noise, outliers, and sparsity. Point-to-plane ICP incorporates local surface normals and is often more effective in structured scenes dominated by planar surfaces. Hybrid ICP combines the two error terms, allowing the weighting to adapt to scene characteristics. These methods provide a useful way to analyse the effect of LiDAR variability on geometric registration independently of inertial propagation.

A fair comparison across LiDARs also requires careful control of preprocessing and hyperparameters. Range constraints, correspondence thresholds, and convergence criteria must be selected consistently across methods and environments. In this way, the focus remains on how sensor characteristics influence local registration and odometry rather than on implementation-specific tuning artefacts.

The broader implication of this subsection is that multimodal localization depends not only on combining different sensor categories, but also on understanding variability within each category. LiDARs differ in resolution, scan regularity, field of view, and effective sampling pattern. These differences affect localization performance, portability, and deployment suitability. A dome-shaped low-cost sensor may provide highly competitive performance in some indoor or cluttered conditions, while a long-range solid-state sensor may be more advantageous in open spaces, and a spinning LiDAR may remain more compatible with pipelines developed around structured scan assumptions. Consequently, LiDAR-based localization in robotic systems must be understood as a sensor-dependent problem rather than as a modality-independent one.

3.2 Seamless Outdoor–Indoor Positioning with GNSS/UWB/IMU Fusion

Localization across outdoor–indoor transitions remains challenging because GNSS, UWB, and inertial pedestrian dead reckoning are complementary yet individually

fragile under signal blockage, multipath, non-line-of-sight propagation, and drift [22; 27; 28; 114]. Outdoors, GNSS provides globally referenced positions but suffers in urban areas from signal blockage, multipath, and dilution of precision near buildings [115]. Indoors, GNSS is largely unavailable; UWB can deliver accurate local positioning where anchors are deployed, but offers only local coverage and is sensitive to non-line-of-sight bias [116; 29]. Inertial sensing provides high-rate self-contained motion tracking, yet pure integration drifts over time [117; 118]. In practice, maintaining continuous localization as a user traverses doorways, corridors, and urban canyons therefore requires principled fusion of these modalities together with environmental knowledge that constrains physically plausible motion.

3.2.1 Proposed Pedestrian Positioning System

The proposed pedestrian positioning system, shown in Fig. 11, combines GNSS, UWB, and IMU-based pedestrian dead reckoning within a unified wearable architecture. A chest-mounted platform carries a Raspberry Pi 5, a GNSS module, an Xsens MTi-300 IMU, and a Decawave DWM1001 UWB tag. The system is designed for scenarios such as industrial campuses or urban research sites, where a user moves intermittently between open-sky outdoor areas and an indoor building instrumented with UWB anchors.

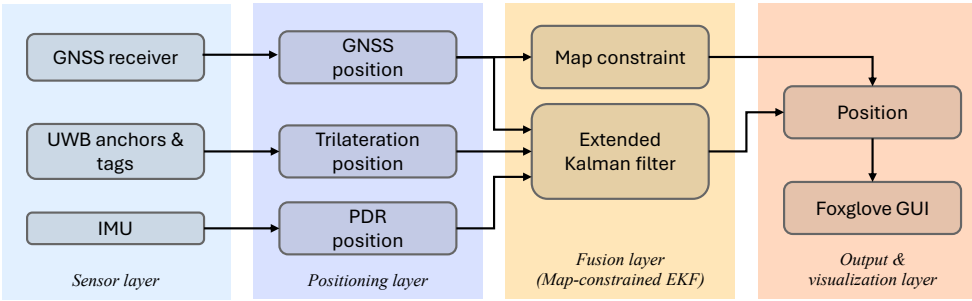
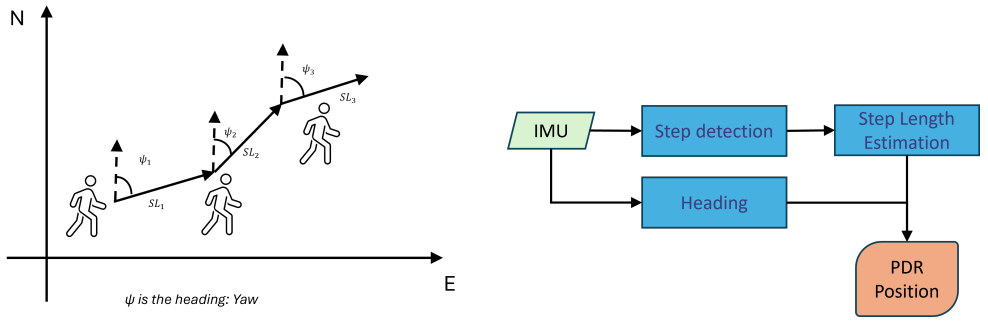


Figure 11. Proposed evaluation scheme in the seamless outdoor-indoor fusion localization study

At the sensing level, GNSS provides outdoor absolute position information, UWB provides indoor absolute position information through trilateration, and the IMU provides the basis for pedestrian dead reckoning. All measurements and estimates are represented in a local East–North–Up frame anchored to the test site. In this architecture, chest-mounted IMU-based Pedestrian Dead-Reckoning (PDR) forms the motion backbone, while GNSS and UWB supply environment-dependent absolute corrections.

The PDR front-end operates in a step-triggered manner, shown in Fig 12. Steps are detected from the band-pass filtered dynamic acceleration magnitude within the

normal walking frequency range. Step length is estimated using a Weinberg-style relation driven by the peak–trough amplitude of the walking cycle, and heading is obtained from the IMU and magnetometer orientation estimate. The resulting horizontal displacement increment is then expressed in the local frame. This formulation is particularly suitable for pedestrian positioning because it limits the accumulation of drift over short walking horizons while maintaining sufficiently high temporal responsiveness for fusion back-ends.



(a) (b)

Figure 12. PDR scheme.

GNSS and UWB serve distinct but complementary roles. GNSS provides globally referenced absolute updates outdoors. UWB provides local ranging information indoors, where surveyed anchors are available. The UWB deployment used six UWB nodes in total, as summarized in Table 3. Four nodes were deployed as static reference nodes around the outdoor–indoor transition area and are reported using WGS84 latitude–longitude coordinates, while the remaining two nodes were used as mobile or movable tags without fixed coordinates across all experiments. One mobile tag was mounted on the chest-worn positioning platform together with the IMU and GNSS receiver, and the other movable tag was placed at experiment-dependent positions. All UWB nodes were placed at approximately 1.5 m above the ground or floor level. Since the static reference nodes and mobile tags were deployed at approximately the same height, the UWB geometry mainly constrained the horizontal position, while the vertical component was weakly observable from UWB ranging alone. Therefore, the positioning analysis focuses on the horizontal position, and the vertical component is treated as a deployment parameter rather than a primary estimated state. In the proposed system, indoor absolute positions are produced by a UWB trilateration node and then fused together with PDR increments. The system therefore avoids relying on a single sensing modality across all environments and instead adapts the source of absolute correction to the operating context.

A distinctive feature of the framework is the use of a lightweight map-based feasibility constraint derived from OpenStreetMap building footprints. At runtime,

Table 3. UWB node deployment used in the outdoor–indoor positioning experiment. Nodes 1–4 were deployed as static reference nodes and are reported using WGS84 latitude–longitude coordinates. The chest-mounted node moved with the pedestrian platform, while the auxiliary movable node had experiment-dependent positions.

Node	Role	Latitude	Longitude
Node 1	Static reference	60.4451347746	22.3150171616
Node 2	Static reference	60.4451876168	22.3150951445
Node 3	Static reference	60.4452319196	22.3149719963
Node 4	Static reference	60.4451783693	22.3148938767
Chest-mounted node	Mobile tag	Trajectory-dependent	Trajectory-dependent
Auxiliary movable node	Movable tag	Experiment-dependent	Experiment-dependent

building polygons are queried and treated as environmental priors. Most building interiors are considered non-navigable, while the designated UWB-instrumented building is explicitly allowed as a free-space region. This simple prior introduces a practical form of environmental awareness into the fusion process. It prevents physically implausible corrections, such as GNSS updates that spuriously place the user inside a non-accessible building near a facade, while still allowing motion within the indoor UWB-covered structure.

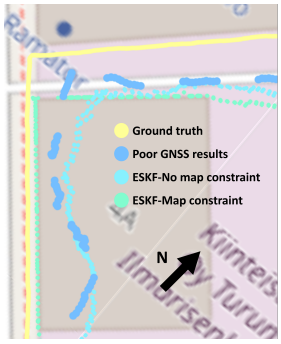
The positioning framework is implemented in ROS 2 and runs in real time on the wearable platform, shown in Fig. 13c. The ROS 2 implementation handles data acquisition, time synchronization, coordinate conversion, trilateration, state estimation, and logging. For visualization and monitoring, the system uses Foxglove to display the fused trajectories, raw GNSS positions, UWB updates, map overlays, and estimator-specific outputs. This unified interface is particularly useful around transition regions, where the behaviour of the estimator can be assessed directly during doorway crossing and building-entry sequences.

The proposed system is therefore not simply a loose combination of sensors, but a structured positioning architecture in which the motion model, absolute measurements, and environmental feasibility constraint are designed to work together. The IMU stabilizes short-horizon motion, GNSS and UWB provide complementary absolute references, and map information suppresses physically implausible updates near transitions. This design forms the basis for the comparative study of the three probabilistic fusion back-ends.

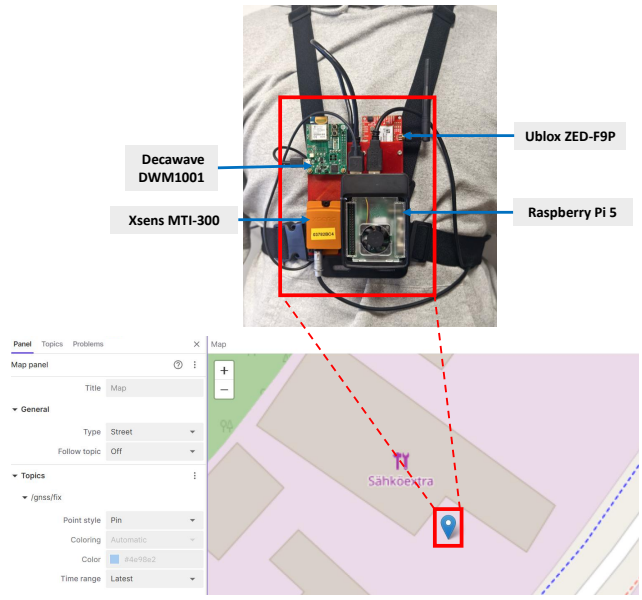
3.2.2 Fusion back-ends: EKF, FGO, and PF

To compare filtering, smoothing, and sampling paradigms under consistent sensing conditions, the system implements three probabilistic back-ends: a planar error-state extended Kalman filter, a sliding-window factor graph optimizer, and a particle filter. All three methods share the same chest-mounted PDR front-end and consume

(a) Building polygons



(b) Map constraint



(c) Chest-Mounted Sensor Platform

Figure 13. Overview of map constraints and the chest-mounted sensor platform.

the same absolute position updates from GNSS and UWB trilateration. This common front-end ensures that the differences observed among the methods originate primarily from the estimator back-end rather than from the sensing or preprocessing pipeline.

The error-state extended Kalman filter is designed as a lightweight recursive estimator, shown in Algorithm 1. Instead of integrating raw inertial acceleration directly, it treats the PDR displacement as the primary motion input and uses GNSS or UWB measurements as position corrections. In this formulation, the state typically contains position, velocity, attitude, and a covariance matrix representing the uncertainty of the error state. During propagation, the filter updates the nominal state using the PDR increment and propagates the covariance through the linearized process model. During correction, absolute measurements are transformed into the local frame and incorporated through the Kalman gain. This design yields high computational efficiency and structured real-time behaviour, making it attractive for wearable or embedded deployments.

The factor graph optimization back-end formulates localization as a local smoothing problem over a sliding window, shown in Algorithm 2. Consecutive states are connected by PDR between-factors, while GNSS and UWB updates are introduced as absolute position factors. A weak anchor prior is commonly added to stabilize

the graph numerically, and window pruning keeps computational growth bounded. Compared with filtering, this approach allows later measurements to refine earlier states within the active window. This can be advantageous in transition regions or during intermittent measurement availability, because the graph can exploit temporal consistency over a short horizon rather than committing irreversibly to each recursive update.

The particle filter represents the posterior distribution through a set of weighted particles, shown in Algorithm 3. In this implementation, particles are propagated when steps are detected, using the same PDR displacement increments that drive the other back-ends, but with stochastic perturbations added to represent uncertainty. When an absolute position fix becomes available, the weight of each particle is updated using the GNSS or UWB measurement likelihood. Degeneracy is monitored through the effective sample size, and resampling is triggered when the particle set becomes overly concentrated. This sampling-based formulation is especially useful when the posterior is strongly nonlinear, multimodal, or affected by non-Gaussian error characteristics, which can occur in challenging GNSS or UWB conditions.

The parameters of the above algorithms are shown in Table 4.

The map-based feasibility constraint is implemented consistently across all three back-ends, but its practical role differs slightly in each case. In the ESKF, an implausible GNSS update can be rejected or projected to the nearest admissible region before correction. In the factor graph, the constraint can be expressed as a feasibility-aware penalty or gating mechanism that discourages state estimates inside forbidden regions. In the particle filter, particles violating the building-occupancy rule can be assigned near-zero likelihood. Although these mechanisms differ operationally, they all serve the same purpose: to regularize the solution near building boundaries and to reduce the influence of multipath-driven or otherwise physically implausible absolute measurements.

From a methodological standpoint, the three back-ends reflect different philosophies of sensor fusion. The ESKF emphasizes computational efficiency and a tightly managed prediction–update cycle. The factor graph optimizer emphasizes local consistency and delayed refinement within a sliding horizon. The particle filter emphasizes representational flexibility in the face of strong nonlinearities or ambiguous transition conditions. Their inclusion in the same system makes it possible to examine how estimator choice influences seamless outdoor–indoor positioning when the sensing and motion front-end remain otherwise unchanged.

Algorithm 1 PDR-driven ESKF

State: $(\mathbf{p}_{\text{est}}, \mathbf{v}_{\text{est}}, q_{\text{est}}, \mathbf{P})$
Error: $[\delta\mathbf{p}, \delta\mathbf{v}, \delta\boldsymbol{\theta}]$, $\mathbf{P} \in \mathbb{R}^{9\times 9}$
Init from first accepted absolute fix
for each IMU/PDR sample **do**
    $\Delta\mathbf{p} \leftarrow \mathbf{p}^{\text{PDR}} - \mathbf{p}^{\text{PDR}}_{\text{prev}}$
    $\mathbf{p}_{\text{est}} \leftarrow \mathbf{p}_{\text{est}} + \Delta\mathbf{p}$; $q_{\text{est}} \leftarrow q^{\text{PDR}}$
    $\mathbf{P} \leftarrow \mathbf{F}\mathbf{P}\mathbf{F}^{\top} + \mathbf{L}\mathbf{Q}\mathbf{L}^{\top}$
end for
for each absolute fix (GNSS/UWB) **do**
    $\mathbf{z} \leftarrow \text{geodetic2enu}(\mathbf{z}_{\text{LLA}})$
    $\mathbf{K} \leftarrow \mathbf{P}\mathbf{H}^{\top}(\mathbf{H}\mathbf{P}\mathbf{H}^{\top} + \sigma^2_{\text{sensor}}\mathbf{I})^{-1}$
    $\delta\mathbf{x} \leftarrow \mathbf{K}(\mathbf{z} - \mathbf{p}_{\text{est}})$
    $\mathbf{p}_{\text{est}} \leftarrow \mathbf{p}_{\text{est}} + \delta\mathbf{p}$; $\mathbf{v}_{\text{est}} \leftarrow \mathbf{v}_{\text{est}} + \delta\mathbf{v}$
    $q_{\text{est}} \leftarrow \text{Exp}(\delta\boldsymbol{\theta}) \otimes q_{\text{est}}$
    $\mathbf{P} \leftarrow (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}$
end for

Algorithm 2 Sliding-window FGO

Node: $\mathbf{T}_k = [x_k, y_k, z_k, \psi_k]^{\top}$, window W
Wait first absolute fix; add prior $\mathbf{r}_{\text{prior}} = \mathbf{T}_0 - \hat{\mathbf{T}}_0$
for each confirmed step k **do**
    $\Delta\mathbf{u}_k = [\Delta x, \Delta y, \Delta z, \Delta\psi]^{\top}$
    Add node guess $\mathbf{T}_k \leftarrow \mathbf{T}_{k-1} \oplus \Delta\mathbf{u}_k$
    Add Between: $\mathbf{r}_{\text{PDR}} = \text{between}(\mathbf{T}_{k-1}, \mathbf{T}_k) - \Delta\mathbf{u}_k$
    if absolute fix available **then**
        Add Position: $\mathbf{r}_{\text{abs}} = [x_k, y_k, z_k]^{\top} - \hat{\mathbf{p}}_k$
    end if
    if nodes $> W$ **then**
        Prune stale factors; weak-anchor oldest node
    end if
    Optimize: $\min \sum \|\mathbf{r}_{\text{prior}}\|^2 + \sum \|\mathbf{r}_{\text{PDR}}\|^2 + \sum \|\mathbf{r}_{\text{abs}}\|^2$
end for

Algorithm 3 Step-driven Particle Filter

Particle: $\mathbf{x}^{(i)} = [x, y, z, \psi]^\top$, weights $w^{(i)}$, N
Init at first absolute ENU fix; $w^{(i)} \leftarrow 1/N$
for each confirmed step k **do**
  $\Delta\mathbf{p} \leftarrow \mathbf{p}^{\text{PDR}} - \mathbf{p}^{\text{PDR}}_{\text{prev}}$, $\Delta\psi \leftarrow \text{wrap}(\psi^{\text{PDR}} - \psi^{\text{PDR}}_{\text{prev}})$
  for $i \leftarrow 1 \rightarrow N$ **do**
    $\mathbf{p}^{(i)} \leftarrow \mathbf{p}^{(i)} + \Delta\mathbf{p} + \boldsymbol{\epsilon}_p$
    $\psi^{(i)} \leftarrow \text{wrap}(\psi^{(i)} + \Delta\psi + \epsilon_\psi)$
  end for
  if absolute fix $\hat{\mathbf{p}}$ available **then**
    $d_M^{2(i)} \leftarrow (\mathbf{p}^{(i)} - \hat{\mathbf{p}})^\top \mathbf{R}^{-1} (\mathbf{p}^{(i)} - \hat{\mathbf{p}})$
    $w^{(i)} \leftarrow w^{(i)} \exp(-\frac{1}{2} d_M^{2(i)})$; normalize
    $\text{ESS} \leftarrow 1/\sum_i (w^{(i)})^2$; resample if $\text{ESS} < \tau N$
  end if
  $\hat{\mathbf{x}} \leftarrow \sum_i w^{(i)} \mathbf{x}^{(i)}$
end for

Table 4. Key back-end parameters used in the released ROS 2 implementations.

Item	ESKF	FGO	PF
Model params	$var_f = 1$, $var_\omega = 1$	$\sigma_{\text{odom},xy} = 0.25$m, $\sigma_{\text{odom},z} = 0.4$m, $\sigma_{\text{odom},\psi} = 10^\circ$	$\sigma_{\text{prop},xy} = 0.18$m, $\sigma_{\text{prop},z} = 0.05$m, $\sigma_{\text{prop},\psi} = 6^\circ$
Measurement noise	$var_{\text{GNSS}} = 10$ (std≈3.16m), $var_{\text{UWB}} = 0.25$ (std=0.5m)	GNSS: $\sigma_{xy} = 3$m, $\sigma_z = 5$m; UWB: $\sigma_{xy} = 1$m, $\sigma_z = 1$m	GNSS: $\sigma_{xy} = 3$m, $\sigma_z = 5$m; UWB: $\sigma_{xy} = 1$m, $\sigma_z = 1$m
Backend hyper-params	$N_{\text{skip}} = 5$ (UWB); skip GNSS once UWB available	$W = 80$; iters=5; anchor: $\sigma_{xy} = 2$m, $\sigma_z = 3$m, $\sigma_\psi = 45^\circ$; Huber $\delta = 1$	$N = 250$; resample if ESS $< 0.5N$

3.2.3 Experimental Setting and Visualization

The experimental setting is designed to evaluate the positioning framework in indoor, outdoor, and outdoor–indoor conditions under a consistent hardware and software configuration. The wearable platform integrates the chest-mounted IMU, GNSS receiver, UWB tag, and embedded computing unit. All sensor data are acquired and synchronized through ROS 2, and the outputs of the three fusion back-ends are published using consistent message interfaces.

The test environment consists of a campus-like outdoor area connected to an instrumented indoor laboratory building. This arrangement supports three represen-

tative scenarios. The first is an indoor experiment, where UWB and PDR are used together and the trajectory remains inside the anchor-equipped building. The second is an outdoor experiment, where GNSS and PDR are used together in open or semi-urban outdoor space. The third is a seamless outdoor–indoor experiment, in which the pedestrian moves from outdoor GNSS coverage into the UWB-covered indoor building and back across transition zones.

The ground-truth configuration is adapted to the measurement conditions of each scenario. Indoors, a motion-capture system provides high-precision reference trajectories aligned to the same local coordinate frame. Outdoors, a Real Time Kinematic capable GNSS reference provides accurate trajectory ground truth. For the seamless trial, the reference trajectory is assembled consistently across the two environments, allowing continuous accuracy and continuity assessment through transition regions.

The software environment is based on Ubuntu 20.04 and ROS 2. The three estimators operate on the same ROS 2 sensing streams, and their outputs can be logged, replayed, and visualized uniformly. This ensures that the comparison remains controlled not only at the algorithmic level but also at the middleware and systems level. The implementation is intended for real-time operation on an embedded wearable platform rather than offline post-processing only.

Visualization is performed through a unified Foxglove dashboard [119]. The interface displays raw GNSS fixes, UWB positions, fused trajectories from the ESKF, FGO, and PF pipelines, map overlays, and PDR step events. This dashboard plays a particularly important role in transition scenarios, where estimator behaviour around doorway crossings, facade-adjacent GNSS degradation, and intermittent UWB visibility can be examined in a temporally aligned manner. It also supports offline inspection of estimator consistency and sensor interaction.

A representative visualization setup includes building polygons obtained from OpenStreetMap, the designated feasible indoor region, and the trajectories produced by the three estimators. The pedestrian platform, together with the map constraint, thus forms a complete closed-loop evaluation environment: sensing, state estimation, environmental prior, and visualization are all integrated within the same ROS 2-based framework.

4 Resilient Localization in Distributed Robotic Systems

Distributed robotic systems are increasingly expected to operate across heterogeneous hardware platforms, multiple network domains, and dynamically changing environments. In such systems, resilience is not an auxiliary property but a prerequisite for dependable operation. Communication, computation, and perception may all be distributed across robots, edge devices, and external infrastructure; as a result, the failure of a single software component or edge node can propagate through the system and compromise the availability of higher-level robotic capabilities. Within this dissertation, resilience is therefore treated as a systems capability that complements communication and localization. While the preceding chapters address communication-aware deployment and localization under heterogeneous sensing conditions, the present chapter examines how distributed robotic systems can preserve operational continuity when software nodes fail during execution.

This chapter draws on two complementary lines of work. The first concerns distributed robotic systems in the edge-cloud continuum and the technologies that enable containerized robotic applications to run at the edge or in the cloud. This section is summarized from the corresponding content of Publication I. The second concerns a ROS 2-based multi-robot localization application in which Kubernetes is used to orchestrate distributed error-mitigation and localization nodes under induced failures. This section summarizes the work from Publication VI. The objective of the chapter is not to reproduce the full experimental results of the appended article, but to establish the conceptual background, system architecture, and evaluation logic through which resilience is examined in this dissertation.

4.1 Resilience as a Capability in Distributed Robotic Systems

Robotic systems are more connected, networked, and distributed than ever. Much potential can be found at the intersection of autonomous robots with the edge and cloud domains. This is leading to a myriad of new architectures, technologies, and proposals for robots to leverage the edge-cloud computing continuum to enhance autonomy, drive multi-robot cooperation and human-robot collaboration, and provide additional computational capabilities. At the same time, the increasing connectivity

and distribution of robotic systems introduce new operational dependencies and new failure modes. A distributed system that achieves high functional performance in nominal conditions may still remain brittle if it cannot sustain operation under component failures, intermittent communication, or heterogeneous resource constraints.

In the context of distributed robotics, resilience may be understood as the system-level ability to continue providing an acceptable level of service despite faults, degradation, or changing operational conditions, and to recover from disruptions with limited human intervention. This understanding is broader than algorithmic robustness alone. It encompasses how computation is distributed, how software components are deployed, how failures are detected, and how execution is restored once disruption occurs. In practice, resilience in distributed robotic systems is inseparable from modularity, orchestration, and recoverability. A resilient multi-robot system must not only exchange information across several hosts, but must also tolerate partial failures without collapsing the broader task pipeline.

The edge-cloud continuum provides both opportunities and challenges in this regard. On the one hand, connectivity can be leveraged to build more robust and capable distributed robotic systems, enabling remote visualization and teleoperation, large-scale simulation, fleet management, and computational offloading. On the other hand, the distribution of perception, inference, and control across multiple hosts raises concerns regarding live migration, interoperability, elasticity, decentralization, and enhanced autonomy under unstable connectivity. The ability to distribute workload dynamically can bring more robustness and resilience to multi-robot systems by optimizing computational resources, but these systems exhibit a higher degree of dynamism than more typical cloud or edge clusters. For this reason, resilience must be framed at the architectural level rather than only at the level of individual algorithms.

Within ROS 2-based robotic systems, this problem becomes especially visible in tightly coordinated, multi-agent applications. ROS 2 has become the de facto standard for robotic systems, offering a distributed architecture that is particularly well suited for multi-robot applications. Multi-robot systems have significantly benefited from ROS 2's features, including real-time communication, modular node management, scalability, and seamless inter-robot data exchange. However, deploying ROS 2 applications across heterogeneous hardware platforms remains a complex task, especially in scenarios that require tightly coordinated, multi-agent systems. In such cases, the failure of a single agent can propagate disruptions throughout the system. Ultra-wideband-based relative multi-robot localization is a representative example, because inter-robot dependencies are essential for maintaining accurate relative positioning [39].

4.1.1 ROS 2 for distributed robotics

ROS 2 is a revamped version of the original ROS middleware and has been under development for more than half a decade. Its design principles include distribution, abstraction, asynchrony, and modularity. These characteristics are tightly related to the current reality of connected and decentralized robotic systems, which are moving from single robots to fleets, and from single-host computing to architectures distributed across the edge-cloud continuum. At the same time, this new reality brings additional design requirements. ROS 2 must support built-in security, embedded systems, diverse networking conditions, real-time computing, and product readiness. These requirements correspond closely to the demands of resilient system architectures, which must tolerate alternating network conditions and heterogeneous computing hardware without compromising essential functionality.

A typical ROS 2 system is composed of a number of distributed processes, or nodes, ranging from sensor drivers to algorithm implementations, high-level decision-making modules, and external interfacing or control logic. ROS 2 nodes communicate through several communication patterns: topics as an asynchronous publish-subscribe framework, services as synchronous request-response patterns, and actions as asynchronous interfaces with a request-feedback-response structure well suited to physical actions that require time and interaction with the environment. In addition, ROS 2 provides a series of abstraction layers, with the communication middleware relying on implementations of the industry-standard Data Distribution Service protocol.

These features make ROS 2 a natural substrate for distributed robotics, but they do not by themselves guarantee resilience. In practice, the distributed nature of ROS 2 exposes the system to multiple forms of fragility. Discovery traffic, quality-of-service policies, and middleware behavior interact with the characteristics of wired and wireless networks. The data-intensive nature of DDS, together with the high-density and uncompressed raw data produced by certain sensors, can complicate direct computational offloading from edge to cloud. In multi-host deployments, node failures, host unavailability, and network instability may interrupt the flow of information required by higher-level robotic tasks. Consequently, ROS 2 provides the modular and distributed execution model required for resilient systems, but additional deployment and orchestration mechanisms are needed if resilience is to be achieved at runtime.

From the perspective of this dissertation, ROS 2 is therefore not merely a middleware choice but an enabling layer for resilient distributed execution. It supports the decomposition of robotic applications into modular nodes, the distribution of these nodes across heterogeneous devices, and the explicit management of inter-node communication. This modularity is a precondition for resilience, because software components can only be recovered, restarted, or relocated effectively when they are

sufficiently encapsulated and separated in the first place.

4.1.2 Kubernetes and K3S for edge orchestration

Virtualization technologies are among the core foundations of cloud computing [54]. Whereas virtual machines long provided the backbone of cloud deployment by offering virtualized operating systems, containers introduced a significantly lighter-weight solution. Containers allow an application to run in an environment together with all required dependencies, thereby reducing deployment friction and improving reproducibility across hosts. Docker and Kubernetes have revolutionized virtualization techniques and cloud computing, and both are leading container orchestration tools. Docker provides a runtime and packaging platform through which applications can be built, deployed, and executed as containers. Kubernetes, originally developed by Google, provides a platform for orchestrating such containers.

In robotics, the relevance of containerization extends beyond convenience. Distributed robotic systems operate across heterogeneous hardware platforms and frequently rely on software stacks with tightly coupled dependencies. Containerization offers a practical means of encapsulating these dependencies while preserving portability and repeatability across edge and cloud environments. In this sense, containers support resilience indirectly by making redeployment more deterministic and by reducing the amount of system-specific intervention required when a component fails.

Kubernetes adds a further layer of significance because it automates deployment, scaling, monitoring, and management of containerized applications. In resilience-oriented deployments, this means that orchestration can be used not only to start application components, but also to detect failures and restore service through self-healing mechanisms. Lightweight distributions such as K3S are especially relevant for robotics because they retain the core orchestration semantics of Kubernetes while reducing resource demands, making them suitable for edge clusters built from embedded graphics processing units (GPU) and laptops.

Several studies have explored containerization and orchestration technologies in robotics, emphasizing modularity, portability, and ease of deployment. Existing works have introduced scalable Kubernetes-based frameworks for the Internet of Drones, containerized ROS applications on cloud-server-edge architectures, and edge computing platforms for model predictive control in aerial robots. Other works have proposed container-based methodologies for ROS deployments on heterogeneous and hierarchical edge-cloud architectures, including mixed-criticality orchestration for autonomous systems. These studies collectively underscore the transformative role of containerization and orchestration in improving scalability, efficiency, and reliability in robotic systems.

Nevertheless, a gap remains between architectural proposals and application-level evidence. Much of the existing work concentrates on system design, and many

implementations are based on ROS 1 rather than ROS 2. More importantly, there has been a notable lack of practical investigation into how the deployment of Kubernetes with ROS 2 influences the performance of specific robotic tasks. This is a crucial limitation, because resilience in robotics cannot be established by deployment abstractions alone; it must be demonstrated in the context of concrete robotic capabilities. In this dissertation, that gap is addressed through a fault-oriented case study based on multi-robot relative localization.

4.2 Resilient Multi-Robot Relative Positioning Using Kubernetes and ROS 2

Precise localization serves as a critical foundation in multi-robot systems [39]. Ultra-wideband-based localization systems offer cost-effective and relatively accurate performance, with the potential to fuse ranging data with odometry and other sensor modalities. As such, they form the backbone of many collaborative robotic applications. However, UWB measurements are susceptible to multiple sources of error, including noise, non-line-of-sight interference, and multipath effects, all of which can compromise system accuracy. Recent work has shown that integrating Long Short-Term Memory (LSTM) models with UWB-based localization can significantly improve accuracy by mitigating ranging errors [39; 120]. Yet this improvement introduces an additional systems challenge: once ranging error estimation is distributed across several edge nodes, the failure of one or more nodes can degrade the overall localization pipeline.

This section presents a resilience-oriented system in which a UWB-based multi-robot relative positioning application is deployed on a Kubernetes edge cluster using ROS 2. The core idea is to combine distributed ranging-error mitigation at the edge with centralized relative localization, and to examine whether orchestration can preserve operational continuity when some of the edge-side mitigation nodes fail. The system therefore uses resilience not as an abstract architectural aspiration, but as a measurable property of a concrete robotic workload.

4.2.1 System Architecture and Deployment

Proposed system architecture

The proposed system architecture, shown in Fig. 14, is designed to leverage sensor fusion and distributed computing in order to achieve resilient and accurate relative localization in a multi-robot setting. The architecture integrates UWB ranging for distance measurement, LSTM networks for UWB error estimation, and particle-filter-based relative localization. The overall design follows a worker–master pattern under K3S orchestration.

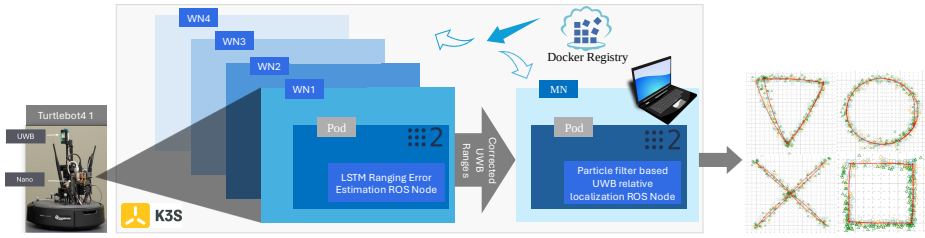


Figure 14. Illustration of the proposed system architecture. A Worker Node (WN) and a Master Node (MN) are managed by K3S. UWB measurements from the robots (WNs) are refined via an LSTM-based Ranging Error Estimation ROS 2 Node and sent to the MN, where a Particle Filter-based ROS 2 Node processes them for relative localization. The ROS 2 nodes run in a container, Pod. The LSTM ranging error estimation programs and particle filter programs are pulled from a private Docker registry to WNs or MN. This setup enhances the resilience and robustness of UWB multi-robot indoor positioning.

In the deployed configuration, each Turtlebot4 is equipped with one UWB transceiver and one NVIDIA Jetson Nano. Each worker node collects UWB ranging data from its local transceiver and runs an LSTM-based ROS 2 node that estimates and corrects ranging errors. The corrected UWB ranges are then published for downstream use. A particle-filter ROS 2 node runs on a laptop acting as the master node, where it processes the corrected UWB data to compute the relative positions of the robots. In this way, the error-mitigation workload is distributed across the edge nodes, while the relative localization back-end remains centralized.

The architectural logic of the system reflects both the strengths and the constraints of edge robotics. On the one hand, distributing the LSTM-based error mitigation across worker nodes makes better use of the available computational resources of the robot-mounted devices. On the other hand, this distribution introduces dependency chains: if one worker-side mitigation node fails, the quality or completeness of the corrected ranging data can be affected. The system therefore provides a suitable basis for investigating whether Kubernetes-based orchestration can recover such nodes automatically and limit the impact of failure on the relative localization task.

K3S orchestration of containerized ROS 2 nodes

Containerization is used to make this deployment operationally feasible. Docker images for the LSTM and particle-filter applications are created and stored in a private Docker registry. This registry serves as a centralized repository for the application images, ensuring secure and efficient distribution of containerized ROS 2 nodes across the edge cluster.

The workflow, shown in Fig. 15 follows a conventional containerization pipeline: Dockerfiles are used to build the images, the images are pushed to the registry, and

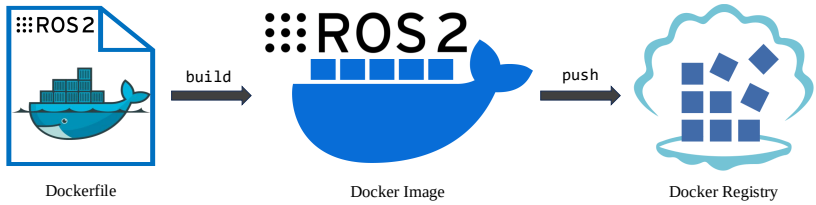


Figure 15. Containerized ROS 2 application workflow

the relevant worker or master nodes pull the required images during deployment. This arrangement supports reproducible deployment and isolates the application runtime from host-specific software variation. For robotics, such consistency is especially important because seemingly minor software mismatches can affect timing, middleware configuration, or driver behavior across heterogeneous hardware.

The particle-filter node is containerized and deployed on the K3S cluster as a ROS 2 application. Functionally, the particle filter subscribes to the raw or corrected UWB ranges and uses these observations together with robot odometry to estimate relative positions. In the implementation adopted for this work, each particle filter propagates its state using odometry and updates the state using range measurements from the UWB transceivers. The state includes the horizontal coordinates of each robot within a shared local reference frame. The deployment setup of the particle filter node is shown as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pf-ros2-deployment
  labels:
    app: pf-ros2
spec:
  selector:
    matchLabels:
      app: pf-ros2
  template:
    metadata:
      labels:
        app: pf-ros2
    spec:
      containers:
        - name: pf-ros2
          image: pf-ros2:ros2core
```

The LSTM nodes are likewise containerized and deployed individually. Because

computational resources are limited in the edge scenario, particularly for machine-learning applications, the LSTM calculation process is divided into multiple nodes. Each LSTM node processes UWB-ranging data associated with its respective transceiver and publishes corrected ranges to a ROS topic. The network architecture consists of two LSTM layers with 50 units each, followed by dropout layers with a 30 percent rate to reduce overfitting, and a final dense layer that outputs a single value representing the estimated ranging error. The model was trained using a separate dataset collected within the same facility, and its effectiveness in mitigating ranging errors had been validated previously. In the present chapter, its importance lies in the fact that it constitutes the workload whose failure and recovery are examined. The deployment setup for the LSTM nodes is illustrated as follows:

```
kind: Deployment
metadata:
  name: uwb-lstm-deploymentxxx
  labels:
    app: uwb-lstmxxx
spec:
  selector:
    matchLabels:
      app: uwb-lstmxxx
  template:
    metadata:
      labels:
        app: uwb-lstmxxx
    spec:
      containers:
        - name: uwb-lstm-ros2core-xxx
          image: uwb-lstm-ros2core:xxx
```

At the orchestration level, the K3S edge cluster manages the deployment and lifecycle of the LSTM and particle-filter nodes. The cluster comprises Jetson Nano worker nodes and a laptop master node, thereby combining resource-constrained robot-mounted devices with a more capable control node. This arrangement is representative of many practical robotic edge deployments, in which not all participating devices have the same computational profile. K3S provides automated management of these containerized components, including failure detection, restart behavior, and load management. In the context of resilience, its relevance lies in the self-healing capability that allows failed application containers to be restored without manual intervention.

The architectural significance of this system extends beyond the particular localization task. It demonstrates how ROS 2 nodes performing estimation and inference

can be treated as orchestrated services in an edge cluster, rather than as monolithic processes tied permanently to a single robot. This perspective is central to the dissertation's broader treatment of deployable robotic capabilities in distributed systems.

4.2.2 Failure-Oriented Evaluation Setup

The resilience of the proposed system is examined through a controlled failure-oriented evaluation in an indoor environment. Rather than treating failure as an incidental condition, the evaluation deliberately induces failures in the LSTM-based error-mitigation nodes during runtime and observes how the orchestrated system responds. This design allows the impact of software-node failure and subsequent automated recovery to be examined in relation to the relative localization task.

The primary evaluation metric is the Absolute Pose Error (APE) of the relative position estimates. An OptiTrack motion-capture system provides ground truth for the relative localization results, and the APE values are computed using the EVO toolchain in Python(`https://github.com/MichaelGrupp/evo`). The choice of APE is appropriate in this context because the central question is whether orchestrated recovery preserves the localization performance of the robotic system when failures occur. The metric therefore links systems-level resilience to application-level outcome.

The experiments are carried out using Turtlebot4 platforms in a controlled indoor setup. The system is implemented on Ubuntu 20.04 with ROS 2 Galactic in Python 3.8. The LSTM network is developed using the PyTorch framework, and K3S version 1.27.4 is used for deployment. Each robot-side worker node consists of a Turtlebot4 equipped with a Jetson Nano featuring 4 GB of RAM, while the master node is a laptop with an AMD Ryzen 7 5800H CPU and 16 GB of RAM. The UWB transceiver is from Qorvo, and the OptiTrack system covers an area of 9 m × 9 m × 5 m. These hardware and software choices are important because they reflect a realistic resource-constrained edge scenario rather than a purely simulated orchestration environment.

A series of failure scenarios is defined in order to examine the system under progressively more severe disruption, shown in Table 5. In addition to a baseline scenario without LSTM correction, the evaluation includes one nominal scenario in which all LSTM pods remain active and several failure-injection scenarios in which one or more LSTM pods are manually terminated during operation. In each case, the particle-filter pod on the master node remains operational, while K3S attempts to restart the failed LSTM pods automatically.

Table 5. Abbreviations and descriptions of experimental setting scenarios.

Scenario	LSTM Pods Running at $t = 0$ s	Failure Event at $t = 15$ s	LSTM Pods Running 5 s After Failure	Purpose
NOLSTM	0	—	0	Pure UWB + PF baseline (no correction)
SALL	5	none	5	Ideal case: full correction, no faults
F1	5	kill 1 pod	5 (after restart)	Single-node fault, tests self-healing
F2	5	kill 2 pods	5 (after restart)	Dual-node fault, moderate stress
F3	5	kill 3 pods	5 (after restart)	Majority failure, high stress
F4	5	kill 4 pods	5 (after restart)	Only one pod initially survives
F5	5	kill all 5 pods	5 (after restarts)	Worst case: complete outage, full recovery required

The baseline scenario, denoted NOLSTM, contains no LSTM pods and therefore represents pure UWB plus particle-filter localization without ranging-error correction. The SALL scenario represents the ideal case in which all LSTM pods are running and no faults are introduced. In scenarios F1 to F5, the system begins with all LSTM pods active, after which one, two, three, four, or all five LSTM pods are deliberately terminated at 15 s. Five seconds after failure, K3S attempts to restore the terminated pods. These scenarios provide a structured way to assess single-node failure, moderate multi-node stress, majority failure, near-total degradation, and complete outage followed by recovery.

The evaluation setup is therefore explicitly failure-oriented rather than nominal-performance-oriented. Its purpose is not simply to ask whether LSTM correction improves localization accuracy under ideal conditions, but whether the overall system can preserve acceptable localization behavior when the distributed mitigation components fail and are subsequently restored. This distinction is essential from the perspective of resilience. A system that performs well only when all components remain continuously available is robust only in a narrow sense; a resilient system must accommodate disruption and recovery as part of its normal operating logic.

5 Research Studies and Results

5.1 Publication I: Distributed Robotic Systems in the Edge-Cloud Continuum with ROS 2: A Review on Novel Architectures and Technology Readiness

Summary. This publication set out to map the emerging field of distributed robotic systems that span the edge–cloud continuum while remaining compatible with ROS 2 as the de facto middleware standard for modern robotics. The study reviewed the technical foundations required for such systems, including DDS-based communication, containerisation, computational offloading. It also examined the practical tooling ecosystem around ROS 2, discussing cloud and simulation platforms, and orchestration or fleet-management frameworks including Kubernetes-based deployments, Open-RMF, Vulcanexus, and Zenoh. The paper therefore served as a structured account of how ROS 2 robotics was being reconfigured by cloud-native methods and by the increasing distribution of computation across heterogeneous infrastructure.

Results and contribution. As a review article, the principal outcome was interpretive rather than experimental. The paper demonstrated that the field had moved beyond generic notions of cloud robotics towards a more technically mature discussion of ROS 2-compliant deployments in which computation, simulation, visualisation, teleoperation, and learning could be distributed across robots, edge devices, and cloud services. It showed that containerisation and orchestration were becoming central design patterns, that robot learning was increasingly tied to cloud simulation and CI/CD workflows, and that decentralised approaches based on distributed ledger technologies were beginning to enter the robotics literature. The paper's central contribution was to provide a structured ROS 2-oriented synthesis of the edge–cloud robotics landscape, to assess the technology readiness of the emerging ecosystem.

Author contribution. The thesis author served as the first author of the publication, led the review process and literature synthesis, structured the paper, and prepared the manuscript in collaboration with the co-authors.

5.2 Publication II: Comparison of Middlewares in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS 2 Systems

Summary. This paper moved from architectural review to controlled experimentation by examining how different networking middlewares affect communication performance in multi-host ROS 2 systems. The study focused on CycloneDDS, MQTT, and Zenoh, and assessed their performance under Ethernet, Wi-Fi, and 4G connections. The experiments were deliberately designed to go beyond single-machine middleware tests by measuring latency and throughput across multiple hosts using ROS messages of different sizes and types, including array and point-cloud messages. In order to connect the networking results to actual robotic behaviour, the study also implemented a real-robot experiment in which a TurtleBot 4 executed repeated square trajectories while velocity commands were transmitted through the respective middleware configurations. Additional discussion was included on CPU usage and on the performance implications of enabling ROS 2 security.

Results and contribution. The results showed that the relative advantage of the communication stack depended on the deployment network. In the reported setup, CycloneDDS performed best under Ethernet, while Zenoh performed better under Wi-Fi and 4G. The throughput results are shown in Fig. 16. The latency results are shown in Table 6. In the real-robot experiment, Zenoh also produced the smallest trajectory drift over the test interval, shown in Fig. 17 and Fig. 18. The paper further reported CPU-usage observations and a limited analysis of the performance impact of enabling ROS 2 security, shown in Table 7 and Table 8. The paper's contribution was therefore twofold. Empirically, it provided a comprehensive multi-host comparison of DDS, MQTT, and Zenoh across edge-to-edge and edge-to-cloud communication settings. Methodologically, it linked latency and throughput measurements to robot-level motion drift, thereby showing that middleware evaluation should be carried out not only at message level but also in terms of its effect on embodied robotic performance.

Author contribution. The thesis author played a leading role and contributed to the study conception and design, participated in material preparation, data collection, and analysis, and wrote the first draft of the manuscript.

Table 6. Mean latency with different network setup

(a) Mean latency (ms) with Ethernet

ROS Message	CycloneDDS	Zenoh-TCP	Zenoh-broker	MQTT-nobroker	MQTT-broker
Array1k	**1.29**	1.98	1.78	89.74	91.01
Array4k	**1.30**	1.97	1.91	86.59	3.87
Array16k	**1.55**	2.36	2.24	86.98	3.58
Array64k	3.35	3.35	**3.27**	3.51	14.77
Array256k	**5.37**	7.42	7.42	10.75	48.50
Array1m	**19.28**	32.88	42.69	136.60	5545.04
Array2m	**37.97**	70.97	70.61	312.89	7468.62
PointCloud512k	**11.16**	20.51	18.47	18.45	87.07
PointCloud1m	**21.95**	37.42	41.24	94.92	1020.70
PointCloud2m	**39.94**	76.78	77.12	540.25	5625.40

(b) Mean latency with Wi-Fi

ROS Message	CycloneDDS	Zenoh-TCP	Zenoh-broker	MQTT-nobroker	MQTT-broker
Array1k	51.40	14.82	**12.58**	92.93	173.62
Array4k	39.84	**32.34**	54.11	101.19	73.64
Array16k	56.73	**48.30**	139.63	101.77	125.69
Array64k	606.50	452.51	326.96	**278.81**	4382.25
Array256k	1574.14	3021.77	2323.91	13538.75	19788.57
Array1m	-	**5960.16**	7599.04	20950.00	-
Array2m	-	**9386.79**	11337.42	-	-
PointCloud512k	-	**5164.90**	5407.28	11307.95	-
PointCloud1m	**3748.00**	5326.69	5657.96	24115.00	-
PointCloud2m	-	9283.26	**8805.96**	-	-

(c) Mean latency with 4G

ROS Message	CycloneDDS	Zenoh-TCP	Zenoh-broker	MQTT-nobroker	MQTT-broker
Array1k	**58.80**	72.11	134.02	279.17	679.91
Array4k	103.75	**102.86**	98.05	228.84	537.87
Array16k	**116.95**	150.81	145.42	220.82	530.20
Array64k	**163.29**	3349.90	434.19	541.76	2222.11
Array256k	-	**7458.61**	7556.25	10736.73	19987.27
Array1m	-	**10086.88**	11811.52	-	-
Array2m	-	15365.29	**10870.59**	-	-
PointCloud512k	-	7357.50	**6591.85**	-	-
PointCloud1m	-	13254.00	**10333.13**	-	-
PointCloud2m	-	14309.50	**11900.27**	-	-

5.3 Publication III: Event-based Sensor Fusion and Application on Odometry: A Survey

Summary. This survey addressed a more specific but technically important localisation question: how event cameras can be used in sensor-fusion pipelines for odometry. Unlike broader event-camera surveys, the paper was explicitly organised around odometry and pose estimation. It began by explaining the advantages of event cameras—high temporal resolution, low latency, wide dynamic range, and resistance to motion blur—and then reviewed the main fusion strategies relevant to odometry.

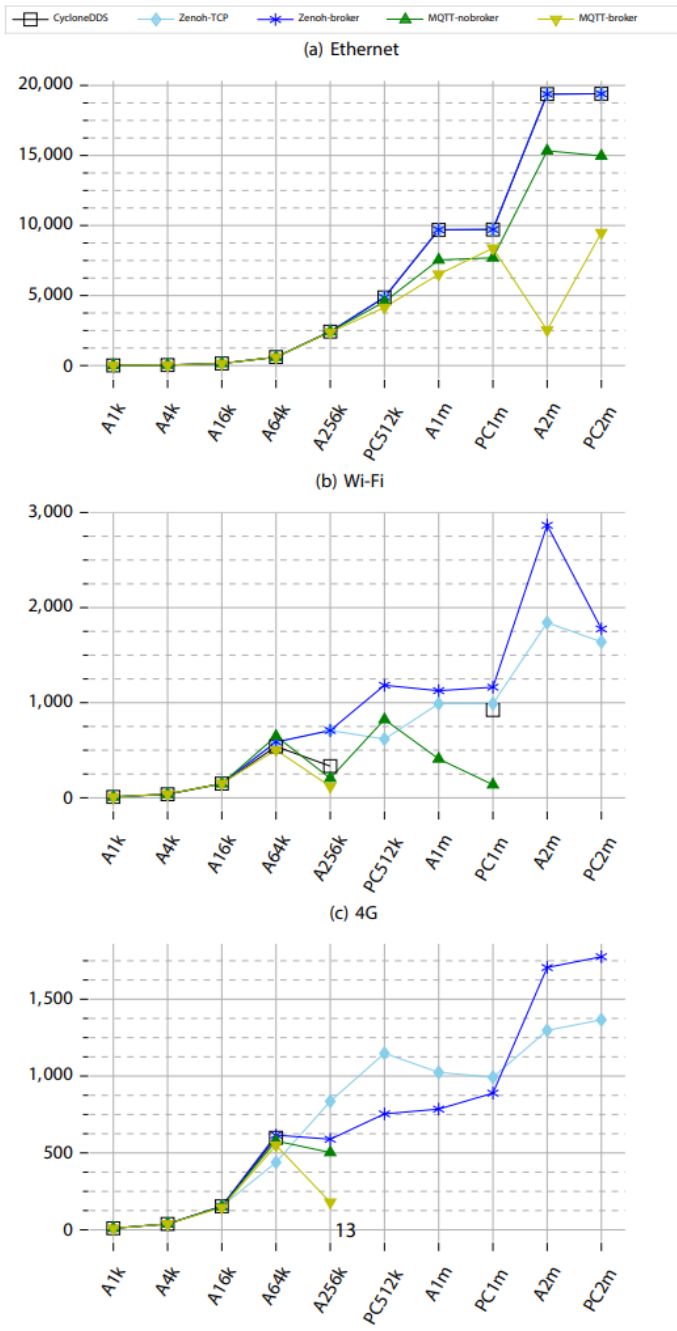


Figure 16. Throughput (Kilobytes/sec) with different network setup. In Subfig (a), since the data difference is very small, the lines of CycloneDDS, Zenoh-TCP, and Zenoh-broker overlap.

These included event-only methods, fusion with frame-based cameras, fusion with

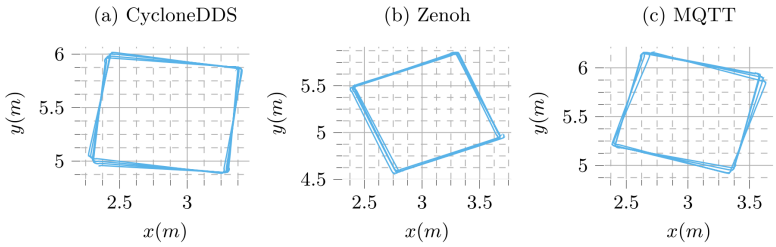


Figure 17. Robot moving trajectory over 96 s of different middleware. The trajectories are for illustration and are not comparable.

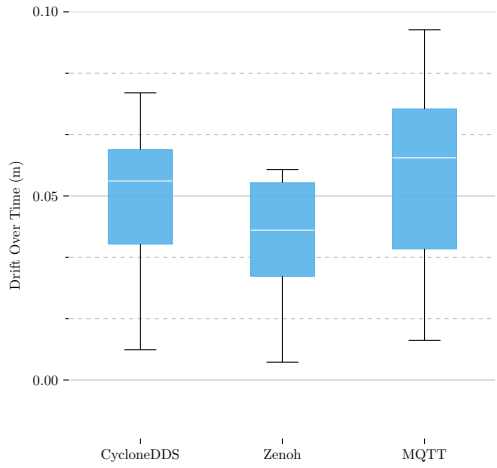


Figure 18. Robot drift over 96 seconds for various middlewares, analyzed using data from 2200 frames per middleware.

IMUs, event-based stereo visual odometry, and early work on event–LiDAR integration.

Results and contribution. The survey showed that event cameras are particularly valuable where conventional visual odometry is compromised by rapid motion, poor lighting, or wide dynamic range, and that fusion with IMUs provides a powerful route to low-latency and drift-reduced motion estimation. It further showed that event-based stereo methods were beginning to mature into credible 3D odometry pipelines, while event–LiDAR fusion remained promising but comparatively immature, with calibration and cross-modal integration still at an early stage. The paper's contribution was to reframe the event-camera literature specifically around odometry, to synthesise the principal fusion strategies with frame cameras, IMUs, stereo setups, and LiDAR, and to identify the most significant technical gaps that remain in event-based sensor fusion for motion estimation.

Author contribution. The author's contribution consisted in repositioning a broad event-camera literature around the narrower and more practically significant

Table 7. CPU usage (%) with different network setup. The data is in the format of (Ethernet_CPU, Wi-Fi_CPU, 4G_CPU).

MSG	CycloneDDS	Zenoh-TCP	Zenoh-broker	MQTT-nobroker	MQTT-broker
Array1k	(0.04, 0.04, 0.04)	(0.05, 0.05, 0.06)	(0.06, 0.05, 0.06)	(0.05, 0.05, 0.05)	(0.05, 0.05, 0.05)
Array4k	(0.04, 0.03, 0.04)	(0.05, 0.05, 0.06)	(0.06, 0.05, 0.06)	(0.05, 0.05, 0.05)	(0.05, 0.05, 0.05)
Array16k	(0.05, 0.04, 0.05)	(0.06, 0.06, 0.06)	(0.06, 0.06, 0.06)	(0.05, 0.05, 0.05)	(0.04, 0.05, 0.06)
Array64k	(0.07, 0.06, 0.06)	(0.06, 0.05, 0.05)	(0.07, 0.05, 0.06)	(0.05, 0.05, 0.06)	(0.05, 0.03, 0.06)
Array256k	(0.19, 0.09, 0.06)	(0.09, 0.05, 0.04)	(0.09, 0.05, 0.05)	(0.08, 0.05, 0.05)	(0.07, 0.04, 0.05)
Array1m	(0.51, 0.08, 0.11)	(0.16, 0.04, 0.03)	(0.18, 0.03, 0.04)	(0.18, 0.13, 0.13)	(0.17, 0.12, 0.14)
Array2m	(0.77, 0.08, 0.19)	(0.30, 0.04, 0.03)	(0.32, 0.04, 0.03)	(0.30, 0.17, 0.19)	(0.20, 0.18, 0.19)
PointCloud512k	(0.30, 0.07, 0.07)	(0.12, 0.06, 0.03)	(0.12, 0.04, 0.04)	(0.12, 0.08, 0.07)	(0.12, 0.09, 0.09)
PointCloud1m	(0.57, 0.09, 0.12)	(0.18, 0.04, 0.03)	(0.19, 0.04, 0.04)	(0.17, 0.12, 0.14)	(0.21, 0.15, 0.14)
PointCloud2m	(0.84, 0.11, 0.19)	(0.32, 0.03, 0.03)	(0.30, 0.03, 0.04)	(0.33, 0.18, 0.20)	(0.22, 0.22, 0.20)

Table 8. Security difference

Security	Mean latency	Throughput	CPU usage
With security	77.35 ms	9701 bytes/sec	0.06 %
Without security	72.11 ms	9804 bytes/sec	0.06 %

theme of odometry, organising the field into clear sensor-fusion categories, and drawing out the implications of those categories for future localisation research.

5.4 Publication IV: Understanding Lidar Variability: A Dataset and Comparative Study Featuring Dome-Shaped, Solid-State, and Spinning Lidars

Summary. This publication examined how localisation performance changes when the sensing platform itself changes. The paper introduced a new multi-lidar dataset in which dome-shaped solid-state, limited-field-of-view solid-state, and spinning lidars were mounted simultaneously on the same robotic platform and recorded in both indoor and outdoor environments with accurate ground truth from motion capture or GNSS/RTK. On this dataset, the study benchmarked a set of state-of-the-art SLAM methods—FAST-LIO2, FASTER-LIO, S-FAST-LIO, FAST-LIO-SAM, and GLIM—and a complementary set of point-cloud matching methods based on point-to-point, point-to-plane, and hybrid ICP. In this way, the work did not merely introduce data; it used that data to investigate the interaction between sensor geometry, scanning pattern, environment type, and odometry algorithm.

Results and contribution. The study showed that sensor modality has a measurable and environment-dependent effect on localisation. The Livox Mid-360 delivered the strongest and most stable overall performance across both SLAM and ICP evaluations, particularly in cluttered or complex scenes where its wider vertical FoV and

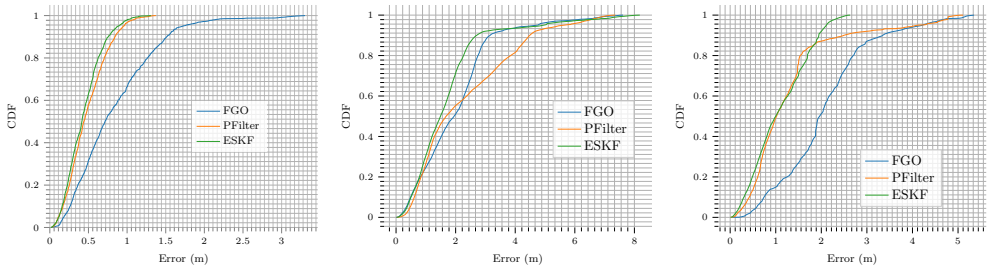
dense coverage proved advantageous. The Livox Avia performed well in long-range outdoor SLAM, especially in road-like environments, but was less effective in local registration tasks. The Ouster spinning lidar yielded competitive and often reliable results in structured scenes, although its advantages were less consistent in semi-structured or heavily cluttered environments. The paper's major contribution was to introduce a new heterogeneous multi-lidar dataset in which dome-shaped, solid-state, and spinning lidars were mounted on the same platform under identical conditions, and to provide a fair benchmark of both SLAM and point-cloud registration methods across these sensing modalities.

Author contribution. The thesis author contributed as is a co-author, and responsible for IMU-GNSS-RTK localization, platform building, and data collection, benchmarking support, results analysis, and writing.

5.5 Publication V: Seamless Outdoor-Indoor Pedestrian Positioning System with GNSS/UWB/IMU Fusion: A Comparison of EKF, FGO, and PF

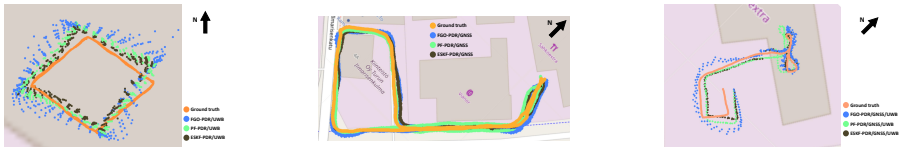
Summary. This publication extended the thesis's localisation strand from robot-centred relative positioning to seamless pedestrian positioning across outdoor–indoor transitions. The study proposed a unified GNSS/UWB/IMU fusion framework in which chest-mounted IMU-based pedestrian dead reckoning provided the motion backbone, GNSS delivered outdoor position updates, UWB supported indoor positioning, and OpenStreetMap building footprints were used as lightweight map constraints to suppress physically implausible solutions. Within this framework, three probabilistic back-ends were implemented and compared under identical sensing conditions: an error-state extended Kalman filter, a sliding-window factor graph optimiser, and a particle filter. The system was realised in ROS 2 on a wearable platform and visualised in Foxglove, with dual ground truth provided by RTK outdoors and motion capture indoors.

Results and contribution. Across all three experimental scenarios—indoor, outdoor, and seamless outdoor–indoor—the error-state Kalman filter provided the most consistent performance. For all scenarios, accuracy is assessed against ground truth using the horizontal error and reported as cumulative distribution functions (CDFs), shown in Fig. 19, together with summary statistics, shown in Table 9. Walking trajectories are shown in Fig. 20. Indoors, it achieved the best reported accuracy, with a median error of 0.415 m and an RMSE of 0.499 m. Outdoors, it again yielded the strongest distribution, with a median error of 1.491 m and an RMSE of 2.186 m. In the seamless transition scenario, it maintained continuity at the doorway while producing the best overall result, with a median error of 0.997 m and an RMSE of 1.248 m. The particle filter was generally the second-best performer, while the factor graph implementation proved less stable in the mixed sensing regime of the study. The pa-



(a) Indoor results (b) Outdoor results (c) Outdoor-to-indoor results

Figure 19. The comparison of CDF under different experimental settings.



(a) Indoor trajectory (b) Outdoor trajectory (c) Outdoor-to-indoor trajectory

Figure 20. Trajectories of the compared methods under different environmental settings.

per's contribution lay in proposing a unified GNSS/UWB/IMU fusion framework for seamless pedestrian positioning, introducing lightweight map-based feasibility constraints derived from building footprints, and providing a controlled comparison of EKF, FGO, and PF within the same wearable ROS 2 implementation.

Table 9. Indoor/Outdoor experiment

	Indoor	**Outdoor**	**Outdoor–Indoor**
	(Mean, Median, RMSE, STD, Max), unit: m		
FGO	(0.836, 0.715, 0.992, 0.535, 3.303)	(1.995, 1.946, 2.389, 1.315, 7.619)	(2.070, 1.962, 2.302, 1.008, 5.351)
Particle filter	(0.491, 0.447, 0.552, 0.253, 1.374)	(2.284, 1.681, 2.816, 1.648, 7.357)	(1.292, 1.017, 1.653, 1.031, 5.108)
ESKF	(0.441, 0.415, 0.499, 0.235, 1.309)	(1.726, 1.491, 2.186, 1.341, 8.181)	(1.085, 0.997, 1.248, 0.617, 2.631)

Author contribution. The thesis author served as the first author of the publication and led the implementation, experimentation, data analysis, and manuscript preparation in collaboration with the co-authors.

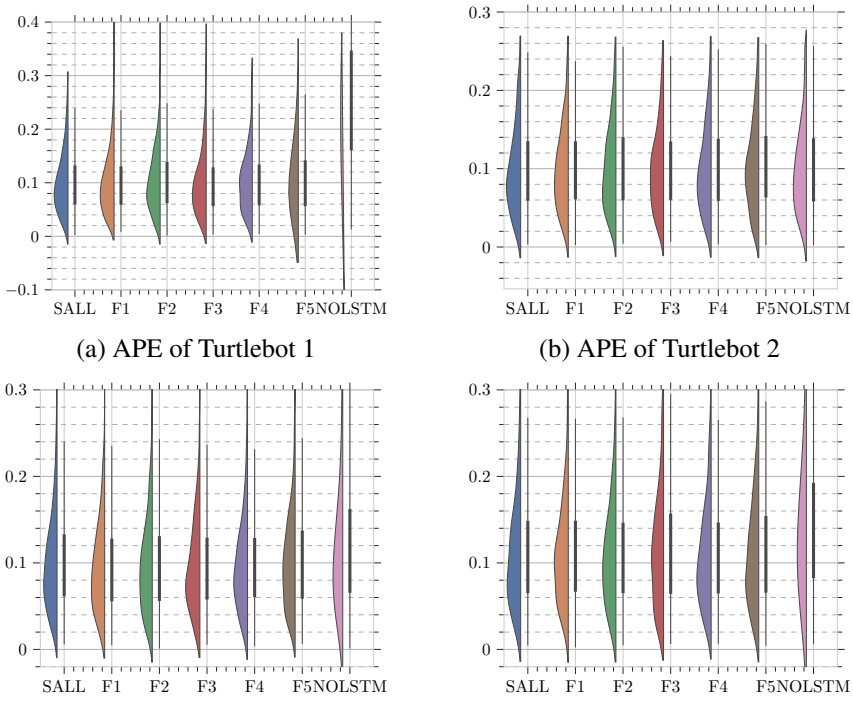
5.6 Publication VI: Enhancing the Resilience of ROS 2-Based Multi-Robot Systems with Kubernetes: A Case Study on UWB-Based Relative Positioning

Summary. This paper addressed resilience directly by asking how a ROS 2-based multi-robot localisation system behaves under node failures when deployed through Kubernetes at the edge. The application case was UWB-based relative positioning, a setting in which inter-robot dependence is intrinsic because distance measurements must be processed cooperatively. The study deployed an edge cluster comprising five Jetson Nano devices and one laptop, orchestrated through K3S. Each robot hosted an LSTM-based UWB ranging-error mitigation node, while the relative pose estimation was performed through a particle-filter node. Controlled failures were then induced by terminating different combinations of LSTM pods, and localisation performance was evaluated from the resulting position errors. This design turned resilience from a general systems claim into a measurable property of a concrete robotic application.

Results and contribution. The results showed that the Kubernetes-orchestrated system remained remarkably stable under failure. The APE values of the relative position results under various experimental settings, as detailed in Table 5, are presented in Figure 21. To provide a concrete comparison of the value differences, Table 10 presents the numerical results for the Root Mean Square Error (RMSE) and its Standard Deviation (STD). From the figure, it is evident that even when LSTM nodes were deliberately stopped, the positioning accuracy remained close to that of the no-failure case and significantly superior to the baseline without LSTM correction. The orchestration layer restored failed components through self-healing, and the overall localisation trajectories(Figure 22) remained largely consistent across the tested scenarios. The paper's principal scientific contribution was to show, through a concrete UWB-based relative positioning case study, that Kubernetes can be used to enhance the resilience of distributed ROS 2 multi-robot applications, and that container orchestration can preserve localisation performance even when edge nodes fail.

Table 10. Root Mean Square Error (RMSE, unit: m) and its corresponding Standard Deviation (STD, unit: m) for different moving robots (TB represents Turtlebot) across various experimental settings.

Scenarios	RMSE/STD			
	TB1	TB2	TB3	TB4
SAll	(0.121/0.064)	(0.122/0.064)	(0.117/0.057)	(0.133/0.070)
F1	(0.116/0.058)	(0.118/0.058)	(0.113/0.057)	(0.132/0.065)
F2	(0.120/0.060)	(0.123/0.062)	(0.116/0.059)	(0.136/0.074)
F3	(0.118/0.062)	(0.125/0.067)	(0.113/0.056)	(0.138/0.071)
F4	(0.118/0.058)	(0.121/0.061)	(0.116/0.059)	(0.131/0.067)
F5	(0.230/0.195)	(0.122/0.060)	(0.118/0.058)	(0.136/0.070)
NOLSTM	(0.661/0.528)	(0.134/0.076)	(0.188/0.127)	(0.255/0.186)



(a) APE of Turtlebot 1

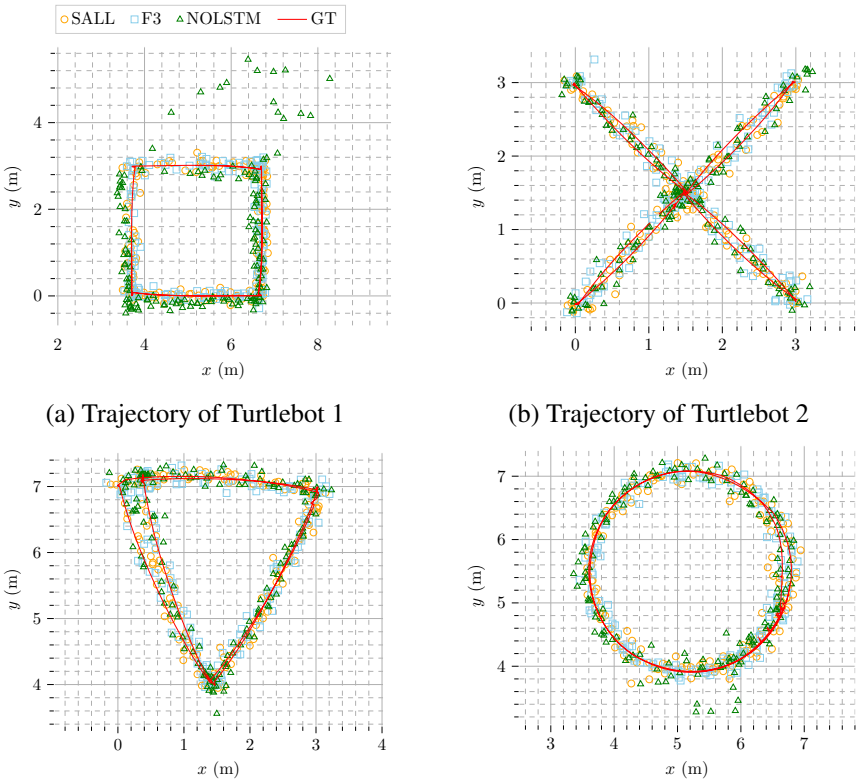
(b) APE of Turtlebot 2

(c) APE of Turtlebot 3

(d) APE of Turtlebot 4

Figure 21. The Absolute Pose Error (APE, unit: m) values of the moving robots under different experimental settings shown in Table 5, Turtlebot 1 to Turtlebot 4, are presented sequentially from left to right.

Author contribution. The thesis author was responsible for conceptualization, methodology, software development, validation, formal analysis, investigation, data processing, original-draft writing, and visualization.



(a) Trajectory of Turtlebot 1

(b) Trajectory of Turtlebot 2

(c) Trajectory of Turtlebot 3

(d) Trajectory of Turtlebot 4

Figure 22. Trajectories of moving robots under different experimental settings, Turtlebot 1 to 4 from left to right.

6 Conclusions

6.1 Summary of the Thesis

This thesis examined robotic capability in distributed systems through three tightly connected dimensions: communication, localization, and resilience. Its central argument is that these are not separable engineering layers, but mutually conditioning properties of the same system. Communication architecture determines whether sensing and coordination remain timely and dependable; localization depends on how heterogeneous measurements are represented, fused, and exchanged; and resilience depends not only on estimator design, but also on how software components are deployed, recovered, and sustained under failure. Across these three themes, the thesis has shown that robust distributed robotics requires co-design across middleware, sensing, estimation, and orchestration rather than isolated optimization of any one component.

Chapter 1 introduced the problem space and formulated the three research questions that guided the thesis. These questions addressed, respectively, the effect of architectural and middleware choices on ROS 2 communication, the role of heterogeneous sensing and multisensor fusion in localization, and the contribution of Kubernetes-orchestrated ROS 2 deployment to resilience in multi-robot relative localization. In doing so, the chapter positioned the dissertation within the broader transition from monolithic robotic systems to distributed, networked, and edge-enabled robotic platforms.

Chapter 2 focused on communication architecture in distributed ROS 2 systems. It first established the broader architectural context of edge–cloud robotics, reviewing the enabling roles of ROS 2, containerization, computational offloading, orchestration, security, and emerging distributed infrastructures. It then examined communication performance empirically through a comparative study of CycloneDDS, MQTT, and Zenoh in multi-host ROS 2 deployments across Ethernet, Wi-Fi, and 4G networks. The chapter showed that communication behaviour in distributed robotic systems is strongly conditioned by deployment context: CycloneDDS performed best in Ethernet environments, whereas Zenoh was more effective under Wi-Fi and 4G, and also yielded the smallest trajectory drift in a real-robot experiment. The chapter therefore established that communication efficiency and reliability in ROS 2 cannot be treated as middleware-invariant properties, but must be analysed in relation

to network conditions and deployment architecture.

Chapter 3 addressed localization in distributed robotic systems through the lens of sensor heterogeneity and multisensor fusion. The chapter first considered event-based sensing as an emerging modality for odometry, showing why event cameras are attractive under high-speed motion, low illumination, and high dynamic range, particularly when fused with conventional cameras, IMUs, and LiDAR. It then examined lidar variability through a comparative dataset and benchmarking study involving dome-shaped solid-state, limited-field-of-view solid-state, and spinning lidars. That analysis showed that sensor geometry and scanning characteristics materially affect SLAM and point-cloud registration performance, and that these effects vary with environmental structure. Finally, the chapter considered seamless outdoor–indoor localization through GNSS/UWB/IMU fusion, comparing ESKF, FGO, and PF back-ends. Taken together, the chapter demonstrated that localization performance emerges from the interaction between sensor characteristics, estimator design, and environmental context, rather than from any universally superior modality or algorithm.

Chapter 4 turned to resilience in distributed robotic systems. Using UWB-based multi-robot relative positioning as a representative application, it studied how containerized ROS 2 nodes deployed on a K3S edge cluster behaved under controlled node failures. LSTM-based UWB ranging error mitigation was distributed across edge nodes, while particle-filter-based localization was executed centrally. By inducing failures in different combinations of the LSTM nodes, the chapter showed that Kubernetes-style orchestration can preserve localization quality through automated recovery and service continuity. The main contribution of the chapter was therefore to move resilience from a purely algorithmic concern to a deployment-level property: the system remained operational not because failure was avoided, but because the software infrastructure could detect, isolate, and recover from failure in a principled way.

Chapter 5 synthesized the findings of the published articles, and summarized the contributions.

6.2 Answers to the Research Questions

RQ1 asked: How do architectural and communication middleware choices shape communication efficiency and reliability in distributed ROS 2 robotic systems across edge-to-edge and edge-to-cloud deployments? The answer developed in this thesis is that communication performance is jointly shaped by architectural decomposition and by middleware–network compatibility. At the architectural level, ROS 2-based distributed robotics increasingly operates across an edge–cloud continuum in which computation, data transport, visualization, orchestration, and security are distributed across heterogeneous resources. At the communication level, the the-

sis showed that no single middleware is optimal in all conditions. CycloneDDS offered the best performance in Ethernet settings, while Zenoh proved more suitable in wireless and wide-area conditions such as Wi-Fi and 4G, including in real-robot operation. The outcome is therefore not a ranking of protocols in the abstract, but a design principle: efficient and reliable ROS 2 communication requires network-aware middleware selection embedded within an architecture that explicitly accounts for locality, bandwidth, latency, and deployment topology.

RQ2 asked: How do heterogeneous sensor characteristics and multisensor fusion strategies jointly determine localization accuracy, robustness, and continuity in robotic systems across diverse environments? The thesis answers this question by showing that heterogeneity is not a complication to be eliminated, but a resource to be exploited. Event cameras contribute high temporal resolution and robustness under fast motion and difficult lighting; LiDARs differ substantially in field of view, scan pattern, density, and environmental suitability; GNSS, UWB, and inertial sensing each provide only partial observability when used alone, but become substantially more reliable when fused. The experimental and review evidence showed that the Livox Mid-360 provided strong and stable performance across several SLAM and ICP settings, that long-range Avia measurements were especially useful in open outdoor conditions, and that ESKF offered the most consistent performance in seamless GNSS/UWB/IMU pedestrian positioning. The broader outcome is clear: localization accuracy, robustness, and continuity depend on matching sensor physics and estimator structure to the operating environment, and on designing fusion methods that exploit complementarity rather than assuming uniform sensor behaviour.

RQ3 asked: How can Kubernetes-orchestrated ROS 2 edge deployment enhance the resilience of UWB-based multi-robot relative localization? The thesis shows that orchestration enhances resilience by transforming failure handling from an ad hoc software concern into an explicit systems capability. In the studied UWB-based relative localization system, LSTM-based ranging correction modules were distributed across edge nodes and exposed to controlled failures. Despite these failures, K3S automatically restarted the affected services, and the resulting localization accuracy remained close to the fully operational case while clearly outperforming the baseline without LSTM correction. The outcome is therefore twofold. First, resilience in distributed robotics is not only a matter of estimator robustness, but also of deployment strategy, node management, and automated recovery. Second, Kubernetes-style orchestration is a practical means of strengthening operational continuity in ROS 2 multi-robot systems, especially when perception and localization services are modularized across heterogeneous edge devices.

Taken together, the answers to the three research questions support a single overarching conclusion. Distributed robotic capability should be understood as a systems property arising from the interaction of communication architecture, sensing and es-

timation, and deployment resilience. Reliable communication without contextual sensing is insufficient; accurate localization without robust deployment is fragile; and resilient deployment without suitable communication and sensing architecture cannot sustain useful autonomy. The principal outcome of the thesis is thus a unified perspective in which communication, localization, and resilience are treated as co-dependent design dimensions of distributed robotic systems.

6.3 Future Work

The most natural continuation of this thesis is the development of a cooperative multi-robot system in which robots relatively localize one another while each robot carries a multimodal sensing suite. Such a system would move beyond the single-application cases examined here and provide a unifying experimental platform for all three capability dimensions studied in this dissertation. Each robot could combine onboard IMU data, LiDAR odometry, visual or event-based perception, UWB ranging, and—when available—GNSS. In that setting, ego-motion estimation would remain onboard and local, while inter-robot relative localization would be constructed from shared range, bearing, and motion constraints. Event cameras would be especially valuable during rapid motion or low-light operation, while heterogeneous LiDARs could contribute complementary geometric coverage depending on whether the fleet was operating in structured indoor spaces, cluttered semi-structured environments, or open outdoor areas.

Methodologically, such a system should adopt a hierarchical fusion architecture. At the first level, each robot would estimate its own local motion using the sensor combination most appropriate to its platform and environment. At the second level, inter-robot UWB ranges, visual detections, LiDAR-based relative pose constraints, or shared landmarks would provide pairwise relative localization. At the third level, the fleet would maintain a distributed pose graph or sliding-window cooperative estimator that fuses absolute anchors, relative measurements, and uncertainty information across the team. This would allow the system to preserve short-term local autonomy even under intermittent communication, while still benefiting from longer-horizon cooperative correction when connectivity is restored. The GNSS/UWB/IMU fusion results of this thesis suggest that such layered estimation is particularly promising for systems that must operate seamlessly across outdoor, indoor, and transitional spaces.

From the perspective of communication and deployment, future work should also pursue a hybrid distributed architecture. Intra-robot communication could continue to rely on native ROS 2 and DDS, especially where local links are stable and low-latency, while inter-robot or edge-facing communication could benefit from middleware better suited to wireless conditions, such as Zenoh. Heavier services—including global map fusion, learned bias correction, long-horizon optimization, and fleet-level

monitoring—could be deployed on a K3S edge cluster, but every robot should retain a minimum self-sufficient localization stack so that temporary network loss does not collapse the entire system. In other words, resilience should be designed as graceful degradation rather than as dependence on uninterrupted connectivity.

Several open problems would need to be addressed for such a system to become practical. Time synchronization across heterogeneous sensors and robots remains difficult, especially when relative localization depends on tightly time-aligned measurements. Extrinsic calibration between multimodal sensors, uncertainty modeling across learned and model-based modules, and the scalable fusion of asynchronous relative constraints all remain unresolved at fleet scale. In addition, the system would need principled fault detection, security-aware communication, and mechanisms for determining when a robot should trust its own sensors more than the information received from peers. These challenges are substantial, but they also define a clear and productive research agenda. The next step after this thesis is therefore not simply to add more sensors or more robots, but to build cooperative multimodal systems in which communication, localization, and resilience are designed from the outset as a single integrated capability.

List of References

[1] Kaiyuan Eric Chen, Yafei Liang, Nikhil Jha, Jeffrey Ichnowski, Michael Danielczuk, Joseph Gonzalez, John Kubiatowicz, and Ken Goldberg. Fogros: An adaptive framework for automating fog robotics deployment. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 2035–2042. IEEE, 2021.

[2] Jeffrey Ichnowski, Kaiyuan Chen, Karthik Dharmarajan, Simeon Adebola, Michael Danielczuk, Vıctor Mayoral-Vilches, Hugo Zhan, Derek Xu, Ramtin Ghassemi, John Kubiatowicz, Ion Stoica, Joseph Gonzalez, and Ken Goldberg. Fogros 2: An adaptive and extensible platform for cloud and fog robotics using ros 2. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5493–5500. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10161307.

[3] Pieter Simoens, Mauro Dragone, and Alessandro Saffiotti. The internet of robotic things: A review of concept, added value and applications. *International Journal of Advanced Robotic Systems*, 15(1):1729881418759424, 2018.

[4] Olimpiya Saha and Prithviraj Dasgupta. A comprehensive survey of recent trends in cloud robotics architectures and applications. *Robotics*, 7(3):47, 2018.

[5] Wang Huaimin, Ding Bo, and Jie Xu. Cloud robotics: a distributed computing view. In *Symposium on Real-Time and Hybrid Systems*, pages 231–245. Springer, 2018.

[6] Wuhui Chen, Yuichi Yaguchi, Keitaro Naruse, Yutaka Watanobe, Keita Nakamura, and Jun Ogawa. A study of robotic cooperation in cloud robotics: Architecture and challenges. *IEEE Access*, 6:36662–36682, 2018.

[7] Mahbuba Afrin, Jiong Jin, Akhlaqur Rahman, Ashfaqur Rahman, Jiafu Wan, and Ekram Hossain. Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(2):842–870, 2021.

[8] Jorge Peña Queralta, Li Qingqing, Tuan Nguyen Gia, Hong-Linh Truong, and Tomi Westerlund. End-to-end design for self-reconfigurable heterogeneous robotic swarms. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2020. doi: https://doi.org/10.1109/DCOSS49796.2020.00052.

[9] Gerardo Pardo-Castellote. Omg data-distribution service: Architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 200–206. IEEE, 2003.

[10] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[11] Stefan Profanter, Ayhun Tekat, Kirill Dorofeev, Markus Rickert, and Alois Knoll. Opc ua versus ros, dds, and mqtt: performance evaluation of industry 4.0 protocols. In *2019 IEEE International Conference on Industrial Technology (ICIT)*, pages 955–962. IEEE, 2019.

[12] osrf. Ros 2 default rmw tsc reports. `https://osrf.github.io/TSC-RMW-Reports/`. [Online].

[13] eProsima. eprosima fast dds performance. `https://www.eprosima.com/index.php/resources-all/performance/eprosima-fast-dds-performance`. [Online].

[14] Christophe Bédard, Pierre-Yves Lajoie, Giovanni Beltrame, and Michel Dagenais. Message flow analysis with complex causal links for distributed ros 2 systems. *Robotics and Autonomous Systems*, 161:104361, 2023. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.

2022.104361. URL https://www.sciencedirect.com/science/article/pii/S0921889022002500.

[15] Wen-Yew Liang, Yuyuan Yuan, and Hsiang-Jui Lin. A performance study on the throughput and latency of zenoh, mqtt, kafka, and dds. *arXiv preprint arXiv:2303.09419*, 2023.

[16] Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Siegwart, et al. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7(66):eabp9742, 2022.

[17] Andrei Cramariuc, Lukas Bernreiter, Florian Tschopp, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, and Cesar Cadena. maplab 2.0–a modular and multi-modal mapping framework. *IEEE Robotics and Automation Letters*, 8(2):520–527, 2022.

[18] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[19] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. *2020 IEEE Int. Conference on Robotics and Automation (ICRA)*, 2020.

[20] Wei Xu and Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2), 2021.

[21] Chunge Bai, Tao Xiao, Yajie Chen, Haoqian Wang, Fang Zhang, and Xiang Gao. Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels. *IEEE Robotics and Automation Letters*, 7(2):4861–4868, 2022.

[22] Shuning Zhang, Hongjing Tang, Liangming Chen, and Yufeng Gao. A seamless pedestrian localization system based on gnss/imu/uwb/map integration. *IEEE Internet of Things Journal*, 2025.

[23] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1), 2020.

[24] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34(6), 2018.

[25] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 6(3), 2021.

[26] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A Al-Ammar, and Hend S Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5):707, 2016.

[27] Vincenzo Di Pietra, Paolo Dabove, and Marco Piras. Loosely coupled gnss and uwb with ins integration for indoor/outdoor pedestrian navigation. *Sensors*, 20(21):6292, 2020.

[28] Debao Yuan, Jian Zhang, Jian Wang, Ximin Cui, Fei Liu, and Yalei Zhang. Robustly adaptive ekf pdr/uwb integrated navigation based on additional heading constraint. *Sensors*, 21(13):4390, 2021.

[29] Jiong Liu, Li Zhang, Jingao Xu, and Jun Shi. Dynamic feasible region-based imu/uwb fusion method for indoor positioning. *IEEE Sensors Journal*, 24(13):21447–21457, 2024.

[30] Itzik Klein. Pedestrian inertial navigation: An overview of model and data-driven approaches. *Results in Engineering*, page 104077, 2025.

[31] Xinyu Hou and Jeroen Bergmann. Rocip: robust continuous inertial position tracking for complex actions emerging from the interaction of human actors and environment. *Applied Intelligence*, 55(6):1–10, 2025.

[32] Zhongliang Deng, Haiming Luo, Xiangchuan Gao, and Peijia Liu. Fusion-based localization system integrating uwb, imu, and vision. *Applied Sciences*, 15(12):6501, 2025.

[33] Sebastian Böhm and Guido Wirtz. Profiling lightweight container platforms: Microk8s and k3s in comparison to kubernetes. In *ZEUS*, pages 65–73, 2021.

[34] Francesco Lumpp, Marco Panato, Franco Fummi, and Nicola Bombieri. A container-based design methodology for robotic applications on kubernetes edge-cloud architectures. In *2021 Forum on specification & Design Languages (FDL)*, pages 01–08. IEEE, 2021.

[35] Francesco Lumpp, Franco Fummi, Hiren D Patel, and Nicola Bombieri. Enabling kubernetes orchestration of mixed-criticality software for autonomous mobile robots. *IEEE Transactions on Robotics*, 40:540–553, 2023.

[36] Francesco Lumpp, Marco Panato, Nicola Bombieri, and Franco Fummi. A design flow based on docker and kubernetes for ros-based robotic software applications. *ACM Transactions on Embedded Computing Systems*, 23(5):1–24, 2024.

[37] Achilleas Santi Seisa, Sumeet Gajanan Satpute, and George Nikolakopoulos. Comparison between docker and kubernetes based edge architectures for enabling remote model predictive control for aerial robots. In *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2022.

[38] Achilleas Santi Seisa, Sumeet Gajanan Satpute, and George Nikolakopoulos. A kubernetes-based edge architecture for controlling the trajectory of a resource-constrained aerial robot by enabling model predictive control. In *2022 26th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, pages 290–295. IEEE, 2022.

[39] Xianjia Yu, Iacopo Catalano, Paola Torrico Morón, Sahar Salimpour, Tomi Westerlund, and Jorge Peña Queralta. Fusing odometry, uwb ranging, and spatial detections for relative multi-robot localization. *arXiv preprint arXiv:2304.06264*, 2023.

[40] Jorge Peña Queralta, Li Qingqing, Zhuo Zou, , and Tomi Westerlund. Enhancing autonomy with blockchain and multi-access edge computing in distributed robotic systems. In *The Fifth International Conference on Fog and Mobile Edge Computing (FMEC 2020)*. IEEE, 2020. doi: https://doi.org/10.1109/FMEC49853.2020.9144809.

[41] Peng Huang, Liekang Zeng, Xu Chen, Ke Luo, Zhi Zhou, and Shuai Yu. Edge robotics: Edge-computing-accelerated multi-robot simultaneous localization and mapping. *IEEE Internet of Things Journal*, 2022.

[42] Eduard Fosch-Villaronga and Christopher Millard. Cloud robotics law and regulation: Challenges in the governance of complex and dynamic cyber–physical ecosystems. *Robotics and autonomous systems*, 119:77–91, 2019.

[43] H Sabireen and V Neelanarayanan. A review on fog computing: Architecture, fog with iot, algorithms and research challenges. *Ict Express*, 7(2):162–176, 2021.

[44] Formant. Formant. `https://formant.io/`. [Online].

[45] AWS. Aws robomaker. `https://aws.amazon.com/robomaker/`. [Online].

[46] Robolaunch. Robolaunch. `https://www.robolaunch.io/`. [Online].

[47] Víctor Mayoral-Vilches, Sabrina M. Neuman, Brian Plancher, and Vijay Janapa Reddi. Robotcore: An open architecture for hardware acceleration in ros 2. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9692–9699. IEEE, 2022. doi: 10.1109/IROS47612.2022.9982082.

[48] Dheera Venkatraman. Rosboard. `https://github.com/dheera/rosboard/`. [Online].

[49] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021. OpenReview.

[50] Open Robotics. Open robotics middleware framework (open-rmf). `https://github.com/open-rmf/rmf`. [Online].

[51] Vulcanexus. Vulcanexus. `https://vulcanexus.org/`. [Online].

[52] S Kekki *et al.* Mec in 5g networks. *ETSI white paper*, 28:1–28, 2018.

[53] Jorge Peña Queralta and Tomi Westerlund. Blockchain-powered collaboration in heterogeneous swarms of robots. *arXiv preprint arXiv:1912.01711*, 2020. Presented at the 2019 Symposium on Blockchain for Robotics and AI Systems, MIT Media Lab.

[54] Claus Pahl. Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3):24–31, 2015.

[55] Steven Armstrong. *DevOps for Networking*. Packt Publishing Ltd, 2016.

[56] Thomas Uphill, John Arundel, Neependra Khare, Hideto Saito, Hui-Chuan Chloe Lee, and Ke-Jou Carol Hsu. *DevOps: Puppet, Docker, and Kubernetes*. Packt Publishing Ltd, 2017.

[57] Jay Shah and Dushyant Dubaria. Building modern clouds: using docker, kubernetes & google cloud platform. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0184–0189. IEEE, 2019.

[58] Golizheh Mehrooz, Emad Ebeid, and Peter Schneider-Kamp. System design of an open-source cloud-based framework for internet of drones application. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pages 572–579. IEEE, 2019.

[59] Lea Matlekovic, Filip Juric, and Peter Schneider-Kamp. Microservices for autonomous uav inspection with uav simulation as a service. *Simulation Modelling Practice and Theory*, 119: 102548, 2022.

[60] Zhuangwei Kang, Kyoungho An, Aniruddha Gokhale, and Paul Pazandak. A comprehensive performance evaluation of different kubernetes cni plugins for edge-based and containerized publish/subscribe applications. In *2021 IEEE International Conference on Cloud Engineering (IC2E)*, pages 31–42. IEEE, 2021.

[61] Kaleem Peeroo, Peter Popov, and Vladimir Stankovic. Exploring the effects of multicast communication on dds performance. In *18th European Dependable Computing Conference (EDCC 2022), Student Forum Proceedings*, 2022. Paper presented at EDCC 2022, Zaragoza, Spain.

[62] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *Proceedings of the 13th International Conference on Embedded Software*, pages 1–10, 2016.

[63] Oasis message queuing telemetry transport (mqtt) tc. `https://www.oasis-open.org/committees/mqtt/`. [Online].

[64] Y Justin Dhas and P Jeyanthi. A review on internet of things protocol and service oriented middleware. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0104–0108. IEEE, 2019.

[65] Roger A Light. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265, 2017.

[66] Eclipse. Zenoh. `https://zenoh.io/`. [Online].

[67] Apex.ai: performance_test. `https://gitlab.com/ApexAI/performance_test`.

[68] Kydos. Minimizing discovery overhead in ros2. `https://zenoh.io/blog/2021-03-23-discovery/#leveraging-resource-generalisation`. [Online].

[69] Ha Sier, Qingqing Li, Xianjia Yu, Jorge Peña Queralta, Zhuo Zou, and Tomi Westerlund. A benchmark for multi-modal lidar slam with ground truth in gnss-denied environments. *Remote Sensing*, 15(13):3314, 2023. doi: 10.3390/rs15133314.

[70] Patrick Lichtsteiner and Tobi Delbruck. A 64x64 aer logarithmic temporal derivative silicon retina. In *Research in Microelectronics and Electronics, 2005 PhD*, volume 2. IEEE, 2005.

[71] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2), 2008. doi: 10.1109/JSSC.2007.914337.

[72] Sier Ha, Honghao Du, Xianjia Yu, Jian Song, and Tomi Westerlund. Enhancing the reliability of lidar point cloud sampling: A colorization and super-resolution approach based on lidar-generated images. *arXiv preprint arXiv:2409.11532*, 2024.

[73] Waseem Shariff, Mehdi Sefidgar Dilmaghani, Paul Kielty, Mohamed Moustafa, Joe Lemley, and Peter Corcoran. Event cameras in automotive sensing: A review. *IEEE Access*, 2024.

[74] David Weikersdorfer and Jörg Conradt. Event-based particle filtering for robot self-localization. In *2012 IEEE Int. Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2012.

[75] David Weikersdorfer, Raoul Hoffmann, and Jörg Conradt. Simultaneous localization and mapping for event-based vision systems. In *Computer Vision Systems*, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39402-7.

[76] Hongjie Liu, Diederik Paul Moeys, Gautham Das, Daniel Neil, Shih-Chii Liu, and Tobi Delbrück. Combined frame-and event-based detection and tracking. In *2016 IEEE Int. Symposium on Circuits and systems (ISCAS)*. IEEE, 2016.

[77] Liyuan Pan, Cedric Scheerlinck, Xin Yu, Richard Hartley, Miaomiao Liu, and Yuchao Dai. Bringing a blurry frame alive at high frame-rate with an event camera. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[78] Ziwei Wang, Liyuan Pan, Yonhon Ng, Zheyu Zhuang, and Robert Mahony. Stereo hybrid event-frame (shef) cameras for 3d perception. In *2021 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[79] Davide Scaramuzza and Zichao Zhang. Visual-inertial odometry of aerial robots. In *Encyclopedia of Robotics*. Springer, 2019. Accepted chapter.

[80] Sherif AS Mohamed, Mohammad-Hashem Haghbayan, Mohammed Rabah, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. Towards dynamic monocular visual odometry based on an event camera and imu sensor. In *Intelligent Transport Systems. From Research and Development to the Market Uptake: Third EAI Int. Conference, INTSYS 2019*. Springer, 2020.

[81] William Chamorro, Joan Solà, and Juan Andrade-Cetto. Event-imu fusion strategies for faster-than-imu estimation throughput. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2023.

[82] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[83] Florian Mahlknecht, Daniel Gehrig, Jeremy Nash, Friedrich M Rockenbauer, Benjamin Morrell, Jeff Delaune, and Davide Scaramuzza. Exploring event camera-based odometry for planetary robots. *IEEE Robotics and Automation Letters*, 7(4), 2022.

[84] Min Seok Lee, Jae Hyung Jung, Ye Jun Kim, and Chan Gook Park. Event-and frame-based visual-inertial odometry with adaptive filtering based on 8-dof warping uncertainty. *IEEE Robotics and Automation Letters*, 2023.

[85] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *Int. Journal of Computer Vision*, 126(12), 2018.

[86] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3d reconstruction with a stereo event camera. In *Proc. of the European conference on computer vision (ECCV)*, 2018.

[87] Yi Zhou, Guillermo Gallego, and Shaojie Shen. Event-based stereo visual odometry. *IEEE Transactions on Robotics*, 37(5), 2021.

[88] Antea Hadviger, Igor Cvišić, Ivan Marković, Sacha Vražić, and Ivan Petrović. Feature-based event stereo visual odometry. In *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021.

[89] Antea Hadviger, Vlaho-Josip Štironja, Igor Cvišić, Ivan Marković, Sacha Vražić, and Ivan Petrović. Stereo visual localization dataset featuring event cameras. In *2023 European Conference on Mobile Robots (ECMR)*. IEEE, 2023.

[90] Kunfeng Wang, Kaichun Zhao, and Zheng You. Stereo event-based visual-inertial odometry. *arXiv preprint arXiv:2303.05086*, 2023.

[91] Peiyu Chen, Weipeng Guan, and Peng Lu. Esvio: Event-based stereo visual inertial odometry. *IEEE Robotics and Automation Letters*, 8(6), 2023.

[92] Zhe Liu, Dianxi Shi, Ruihao Li, and Shaowu Yang. Esvio: event-based stereo visual-inertial odometry. *Sensors*, 23(4):1998, 2023.

[93] Junkai Niu, Sheng Zhong, and Yi Zhou. Imu-aided event-based stereo visual odometry. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11977–11983. IEEE, 2024.

[94] Junkai Niu, Sheng Zhong, Xiuyuan Lu, Shaojie Shen, Guillermo Gallego, and Yi Zhou. Direct visual-inertial odometry with stereo event cameras. *IEEE Transactions on Robotics*, 41:2164–2183, 2025. doi: 10.1109/TRO.2025.3548523.

[95] Kevin Ta, David Bruggemann, Tim Brödermann, Christos Sakaridis, and Luc Van Gool. L2e: Lasers to events for 6-dof extrinsic calibration of lidars and event cameras. In *2023 IEEE Int. Conference on Robotics and Automation (ICRA)*, 2023. doi: 10.1109/ICRA48891.2023.10161220.

[96] Rihui Song, Zhihua Jiang, Yanghao Li, Yunxiao Shan, and Kai Huang. Calibration of event-based camera and 3d lidar. In *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, 2018. doi: 10.1109/WRC-SARA.2018.8584215.

[97] Mingyue Cui, Yuzhang Zhu, Yechang Liu, Yunchao Liu, Gang Chen, and Kai Huang. Dense depth-map estimation based on fusion of event camera and sparse lidar. *IEEE Transactions on Instrumentation and Measurement*, 71, 2022. doi: 10.1109/TIM.2022.3144229.

[98] Hanyu Zhou, Yi Chang, and Zhiwei Shi. Bring event into rgb and lidar: Hierarchical visual-motion fusion for scene flow. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[99] Shihan Peng, Hanyu Zhou, Hao Dong, Zhiwei Shi, Haoyue Liu, Yuxing Duan, Yi Chang, and Luxin Yan. Cosec: A coaxial stereo event camera dataset for autonomous driving. *arXiv preprint arXiv:2408.08500*, 2024.

[100] Peiyu Chen, Weipeng Guan, Feng Huang, Yihan Zhong, Weisong Wen, Li-Ta Hsu, and Peng Lu. Ecmd: An event-centric multisensory driving dataset for slam. *IEEE Transactions on Intelligent Vehicles*, 2023.

[101] Levi Burner, Anton Mitrokhin, Cornelia Fermüller, and Yiannis Aloimonos. Evimo2: an event camera dataset for motion segmentation, optical flow, structure from motion, and visual inertial odometry in indoor scenes with monocular or stereo algorithms. *arXiv preprint arXiv:2205.03467*, 2022.

[102] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3), 2018.

[103] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The Int. Journal of Robotics Research*, 36(2), 2017.

[104] Jiarong Lin and Fu Zhang. Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 3126–3131. IEEE, 2020.

[105] Xiao Zhang, Zhanhong Huang, Garcia Gonzalez Antony, Witek Jachimczyk, and Xinming Huang. Stream-based ground segmentation for real-time lidar point cloud processing on fpga, 2024. URL https://arxiv.org/abs/2408.10410.

[106] Tao Yin, Jingzheng Yao, Yan Lu, and Chunrui Na. Solid-state-lidar-inertial-visual odometry and mapping via quadratic motion model and reflectivity information. *Electronics*, 12(17), 2023. ISSN 2079-9292. doi: 10.3390/electronics12173633. URL https://www.mdpi.com/2079-9292/12/17/3633.

[107] Loris Redovniković, Antun Jakopec, Janusz Bedkowski, and Jurica Jagetić. The affordable diy mandeye lidar system for surveying caves, and how to convert 3d clouds into traditional cave ground plans and extended profiles. *International Journal of Speleology*, 53(3):7, 2025.

[108] Qingqing Li, Xianjia Yu, Jorge Peña Queralta, and Tomi Westerlund. Multi-modal lidar dataset for benchmarking general-purpose localization and mapping algorithms. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3837–3844. IEEE, 2022.

[109] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[110] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[111] Ouster. Ouster sample lidar data. `https://ouster.com/downloads/sample-lidar-data`, 2020. Accessed: 2025-11-27.

[112] Zhiqiang Chen, Yuhua Qi, Dapeng Feng, Xuebin Zhuang, Hongbo Chen, Xiangcheng Hu, Jin Wu, Kelin Peng, and Peng Lu. Heterogeneous lidar dataset for benchmarking robust localization in diverse degenerate scenarios. *The International Journal of Robotics Research*, 0(0): 02783649251344967, 0. doi: 10.1177/02783649251344967.

[113] Hongming Shen, Zhenyu Wu, Yulin Hui, Wei Wang, Qiyang Lyu, Tianchen Deng, Yeqing Zhu, Bailing Tian, and Danwei Wang. Cte-mlo: Continuous-time and efficient multi-lidar odometry with localizability-aware point cloud sampling. *IEEE Transactions on Field Robotics*, 2:165–187, 2025. doi: 10.1109/TFR.2025.3543142.

[114] Qi Liu, Chengfa Gao, Rui Shang, Zihan Peng, Ruicheng Zhang, and Lu Gan. Environment perception based seamless indoor and outdoor positioning system of smartphone. *IEEE Sensors Journal*, 22(17):17205–17215, 2022.

[115] Changhui Jiang, Yuwei Chen, Chen Chen, Jianxin Jia, Haibin Sun, Tinghuai Wang, and Juha Hyyppa. Implementation and performance analysis of the pdr/gnss integration on a smartphone. *GPS Solutions*, 26(3):81, 2022.

[116] Zhiang Jin, Yanjun Li, Zhe Yang, Yufan Zhang, and Zhen Cheng. Real-time indoor positioning based on ble beacons and pedestrian dead reckoning for smartphones. *Applied Sciences*, 13(7): 4415, 2023.

[117] Lei Wu, Shuli Guo, Lina Han, and Cekderi Anil Baris. Indoor positioning method for pedestrian dead reckoning based on multi-source sensors. *Measurement*, 229:114416, 2024.

[118] Jianyu Wang, Jinhao Liu, Xiangbo Xu, Zhibin Yu, and Zhe Li. A yaw correction method for pedestrian positioning using two low-cost mimus. *Measurement*, 217:112992, 2023.

[119] Foxglove. Foxglove. `https://foxglove.dev/`. [Online].

[120] Víctor Miramá, Alfonso Bahillo, Víctor Quintero, and Luis Enrique Díez. Nlos detection generated by body shadowing in a 6.5 ghz uwb localization system using machine learning. *IEEE Sensors Journal*, 2023.